Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра Систем искусственного интеллекта

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Г. М. Цибульский
подпись
«___» _____ 2017 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 — Информационные системы и технологии

Разработка информационной системы процедурной обработки изображений на основе искусственных нейронных сетей

Руководитель _____ ст. преподаватель каф. СИИ А. В. Пятаева
подпись, дата
Выпускник _____ М. А. Ипполитов
подпись, дата
Консультант _____ Д. А. Перфильев
подпись, дата

Красноярск 2017

Продолжение титульного листа бакалаврской работы по теме «Разработка информационной системы процедурной обработки изображений на основе искусственных нейронных сетей»

Нормоконтролер _____ М. А. Аникьева

подпись, дата

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная работа посвящена разработке и исследованию программного обеспечения процедурной обработки изображений с помощью нейронных сетей. Обработка изображений представляет собой область компьютерной графики, исследующая задачи, в которых входные и выходные данные являются изображениями.

На данный момент одной из наиболее актуальных задач искусственного интеллекта является машинное творчество. В этом направлении рассматриваются проблемы написания компьютером музыки, литературных и художественных произведений. Одним из наиболее перспективных направлений машинного творчества является процедурная генерация изображений.

Машинное творчество представляет собой широкое и относительно молодое направление науки об искусственном интеллекте. Процедурная генерация изображений с помощью нейронных сетей, как часть машинного творчества, является актуальным предметом самых разных исследований.

Целью данной работы является разработка информационной системы процедурной обработки изображений на основе искусственных нейронных сетей.

Для достижения указанной цели необходимо решить следующие задачи:

1) произвести анализ предметной области;
2) изучить теоретические основы обработки изображений на основе нейронных сетей;
3) разработать приложение процедурной обработки изображений на основе нейронных сетей;
4) провести тестирование разработанной системы.

Первая глава посвящена описанию основных положений теории нейронных сетей.

Во второй главе описаны методы и алгоритмы обработки изображений. В

ней также говорится о прикладных задачах в области распознавания изображений.

В третьей главе содержится обзор выбранных инструментов для реализации, а также говорится о разработке программного обеспечения, тестировании и экспериментальном исследовании информационной системы процедурной обработки изображений на основе искусственных нейронных сете.

В заключении приводятся основные выводы и результаты работы.

1 Основы теории нейронных сетей

1.1 Искусственный нейрон

Несмотря на существенные различия, отдельные типы нейронных сетей обладают несколькими общими чертами. Во-первых, основу каждой нейронных сетей составляют относительно простые, в большинстве случаев - однотипные, элементы (ячейки), имитирующие работу нейронов мозга [1]. Далее под нейроном будет подразумеваться искусственный нейрон, то есть ячейка нейронной сети. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов - однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон - выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рисунке 1 [2].

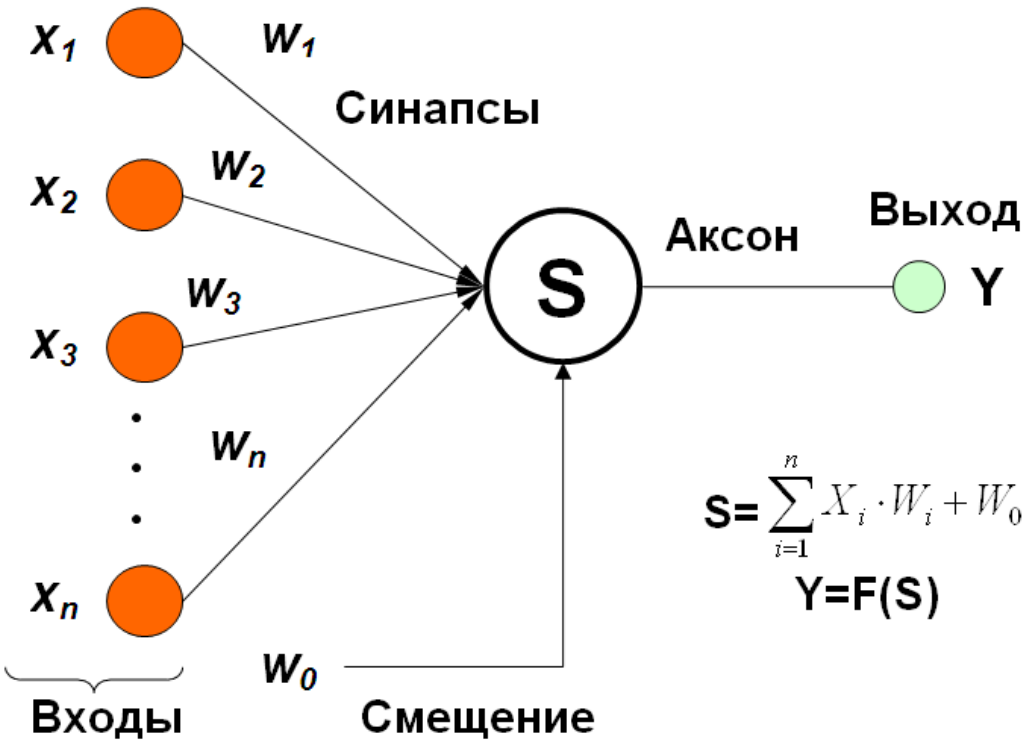

Рисунок 1 — Модель искусственного нейрона

Каждый синапс характеризуется величиной синаптической связи или ее весом, который по физическому смыслу эквивалентен электрической проводимости.

Текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i \quad (1)$$

Выход нейрона есть функция его состояния:

$$y = f(s) \quad (2)$$

Нелинейная функция f называется активационной и может иметь различный вид, как показано на рисунке 2 [2]. Одной из наиболее распространенных, является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид [2]:

$$f(x) = \frac{1}{1+e^{-\alpha x}} \quad (3)$$

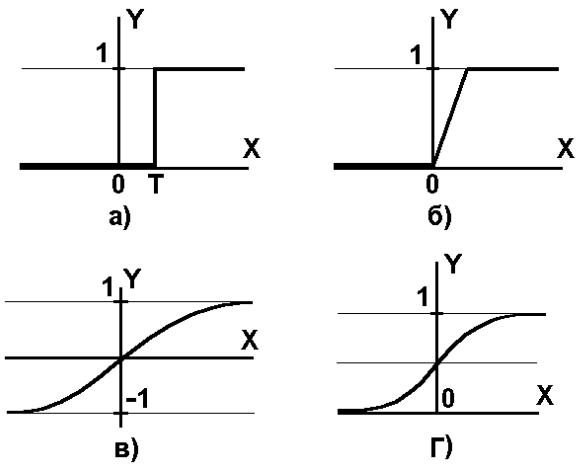

Рисунок 2 — Типы функций активации: а) функция единичного скачка; б) линейный порог (гистерезис); в) сигмоид — гиперболический тангенс; г) логистическая функция

При уменьшении α сигмоид становится более пологим, в пределе вырождаясь в горизонтальную линию на уровне 0,5. При увеличении α сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке $x = 0$. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Одно из ценных свойств сигмоидной функции — простое выражение для ее производной, применение которого будет рассмотрено в дальнейшем.

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x)) \quad (4)$$

Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон [2].

1.2 Архитектуры нейронных сетей

Возвращаясь к общим чертам, присущим всем нейронным сетям, отметим принцип параллельной обработки сигналов, который достигается путем объединения большого числа нейронов в так называемые слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой, причем обработка взаимодействия всех нейронов ведется послойно. Нейронной сетью будем называть структуру, состоящую из связанных между собой нейронов. Нейронные сети могут иметь различные архитектуры. Можно выделить три основных типа нейронных сетей:

- полносвязные сети (рисунок 3-а);
- многослойные сети (рисунок 3-б);
- слабосвязные сети (рисунок 3-в).

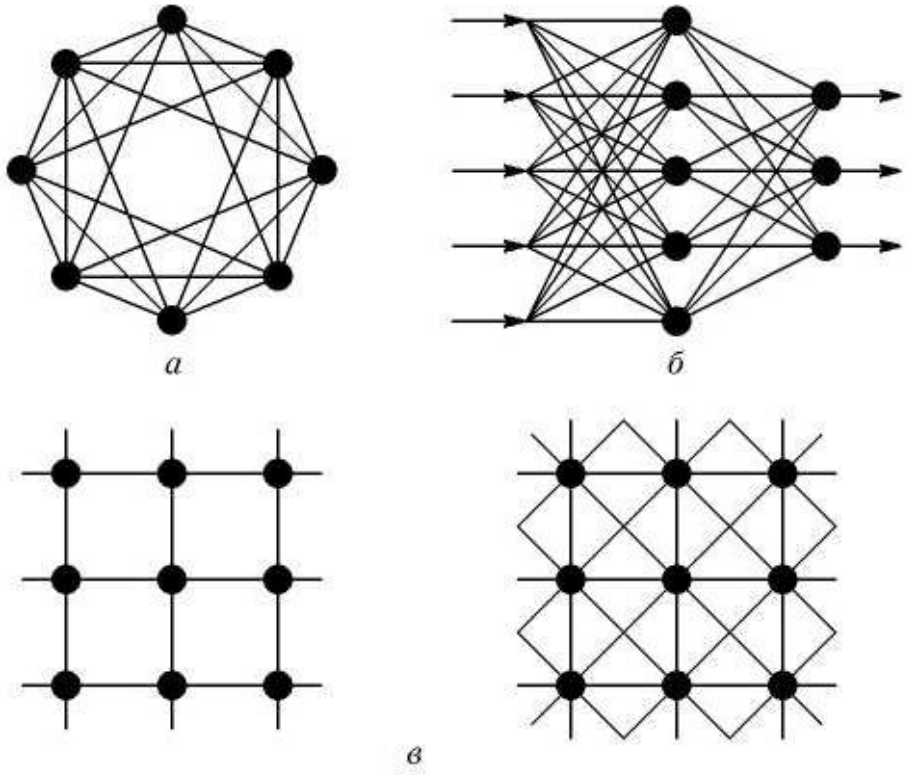

Рисунок 3 — Архитектуры нейронных сетей: а) полносвязная сеть; б) многослойная сеть с последовательными связями; в) слабосвязные сети

В слабосвязных сетях нейроны располагаются в узлах прямоугольной решетки. Каждый нейрон связан с четырьмя или восемью своими ближайшими соседями.

В полносвязной сети каждый нейрон связан со всеми остальными (на входы каждого нейрона подаются выходные сигналы остальных нейронов).

В многослойных сетях нейроны объединяются в слои. Слой - это совокупность нейронов с единым входным сигналом. Внешние входные сигналы подаются на входы нейронов первого слоя, а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько промежуточных (скрытых) слоев. Вход нейронной сети можно рассматривать как выход нулевого слоя вырожденных нейронов. Связи от выходов нейронов некоторого слоя m к входам нейронов следующего слоя $m+1$ называются последовательными.

1.3 Обучение нейронных сетей

Способность к обучению является фундаментальным свойством мозга. В контексте искусственной нейронной сети процесс обучения может рассматриваться как настройка архитектуры сети и весов связей для эффективного выполнения специальной задачи. Обычно нейронная сеть должна настроить веса связей по имеющейся обучающей выборке. Функционирование сети улучшается по мере итеративной настройки весовых коэффициентов. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют определенной системе правил функционирования, сформулированной экспертами.

Для конструирования процесса обучения, прежде всего, необходимо иметь модель внешней среды, в которой функционирует нейронная сеть — знать доступную для сети информацию. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети - какие правила обучения управляют процессом настройки. Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов.

Существуют три парадигмы обучения: "с учителем", "без учителя" и смешанная. В первом случае нейронная сеть располагает правильными ответами на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы как можно более близкие к известным правильным ответам. Усиленный вариант обучения с учителем предполагает, что известна только критическая оценка правильности выхода нейронной сети, но не сами правильные значения выхода. Обучение без учителя не требует знания правильных ответов на каждый пример обучающей выборки. В этом случае раскрывается внутренняя структура данных или корреляции между образцами в системе данных, что позволяет распределить образцы по категориям. При смешанном обучении часть весов определяется посредством обучения с учителем, в то время как остальная получается с помощью самообучения.

Теория обучения рассматривает три фундаментальных свойства, связанных с обучением по примерам: емкость, сложность образцов и вычислительная сложность. Под емкостью понимается, сколько образцов может запомнить сеть, и какие функции и границы принятия решений могут быть на ней сформированы. Сложность образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению. Слишком малое число примеров может вызвать "переобученность" сети, когда она хорошо функционирует на примерах обучающей выборки, но плохо - на тестовых примерах, подчиненных тому же статистическому распределению. Известны 4 основных типа правил обучения: коррекция по ошибке, машина Больцмана, правило Хебба и обучение методом соревнования [3].

1.3.1 Правило коррекции по ошибке

При обучении с учителем для каждого входного примера задан желаемый выход d. Реальный выход сети y может не совпадать с желаемым. Принцип коррекции по ошибке при обучении состоит в использовании сигнала $(d - y)$ для модификации весов, обеспечивающей постепенное уменьшение ошибки. Обучение имеет место только в случае, когда перцептрон ошибается.

1.3.2 Обучение Больцмана

Представляет собой стохастическое правило обучения, которое следует из информационных теоретических и термодинамических принципов. Целью обучения Больцмана является такая настройка весовых коэффициентов, при которой состояния видимых нейронов удовлетворяют желаемому распределению вероятностей. Обучение Больцмана может рассматриваться как специальный случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояний в двух режимах.

1.3.3 Правило Хебба

Самым старым обучающим правилом является постулат обучения Хебба. Хебб опирался на следующие нейрофизиологические наблюдения: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью этого правила является то, что изменение синаптического веса зависит только от активности нейронов, которые связаны данным синапсом.

1.3.4 Обучение методом соревнования

В отличие от обучения Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, при соревновательном обучении выходные нейроны соревнуются между собой за активизацию. Это явление известно, как правило "победитель берет все". Подобное обучение имеет место в биологических нейронных сетях. Обучение посредством соревнования позволяет кластеризовать входные данные: подобные примеры группируются сетью в соответствии с корреляциями и представляются одним элементом.

1.4 Сверточные нейронные сети

Сверточная нейронная сеть является самой мощной технологией глубокого обучения для распознавания изображений. Каждый слой сверточной нейронной сети представляет собой набор плоскостей, состоящих из нейронов. Нейроны одной плоскости имеют одинаковые синаптические (весовые) коэффициенты, ведущие ко всем локальным участкам предыдущего слоя. Каждый нейрон слоя получает входы от некоторой области предыдущего слоя (локальное рецептивное поле), т. е. входное изображение предыдущего слоя сканируется небольшим окном и пропускается сквозь набор синаптических коэффициентов, а результат отображается на соответствующий нейрон

текущего слоя. Таким образом, набор плоскостей представляет собой карты характеристик, и каждая плоскость находит «свои» участки изображения в любом месте предыдущего слоя. Размер локального рецептивного поля выбирается самостоятельно в процессе разработки нейронной сети [4].

В отличие от обычных нейронных сетей [5], в сверточных сетях различают несколько видов слоев в зависимости от их свойств и назначения. Приведем краткое описание каждого из видов слоев:

1) Сверточные слои.

Название слоя произошло от названия математической операции свертки. Пусть есть сигнал $x(t)$ и функция взвешивания $w(t)$, которую еще называют ядро или фильтр. Тогда операция свертки запишется как

$$s(t) = (x * w)(t) = \int x(a)w(t - a)\partial a. \quad (5)$$

Переходя к терминам сверточных нейронных сетей, x — вход слоя, обозначаемый далее X, w — фильтр, обозначаемый K. Рассматривая дискретный случай и ядро с ограниченным носителем, получим $S(i) = \sum_m X(i)K(i - m)$. В большинстве библиотек машинного обучения реализуется несколько другая операция — кросс-корреляция:

$$S(i) = \sum_m X(i + m)K(m) \quad (6)$$

или в двухмерном случае:

$$S(i, j) = \sum_m \sum_n X(i + m, j + n)K(m, n) \quad (7)$$

Основное отличие сверточных слоев от полносвязных состоит в том, что каждый нейрон в нем соединен только с ограниченным числом нейронов из предыдущего слоя. Такой слой рассматривают как совокупность фильтров. Каждому фильтру ставят в соответствие несколько параметров:

а) размер;

б) шаг;

в) набор весов.

Пример реализации свертки показан на рисунке 4. Каждый фильтр в сети реализуется набором нейронов, каждый из которых подсоединен только к своей области видимости — это свойство носит название локальности зоны восприимчивости нейрона [5]. Еще одной важной парадигмой в этой теории является разделение параметров — идея о том, что нейроны, реализующие один фильтр, имеют одинаковые веса. Объединяя два этих концепта в одно, можно сказать, что сверточный слой обладает свойством разреженности связей. В силу того, что размер фильтра обычно на порядки меньше размера входа, требуется гораздо меньше параметров и машинного времени для того, чтобы вычислить выход каждого нейрона.

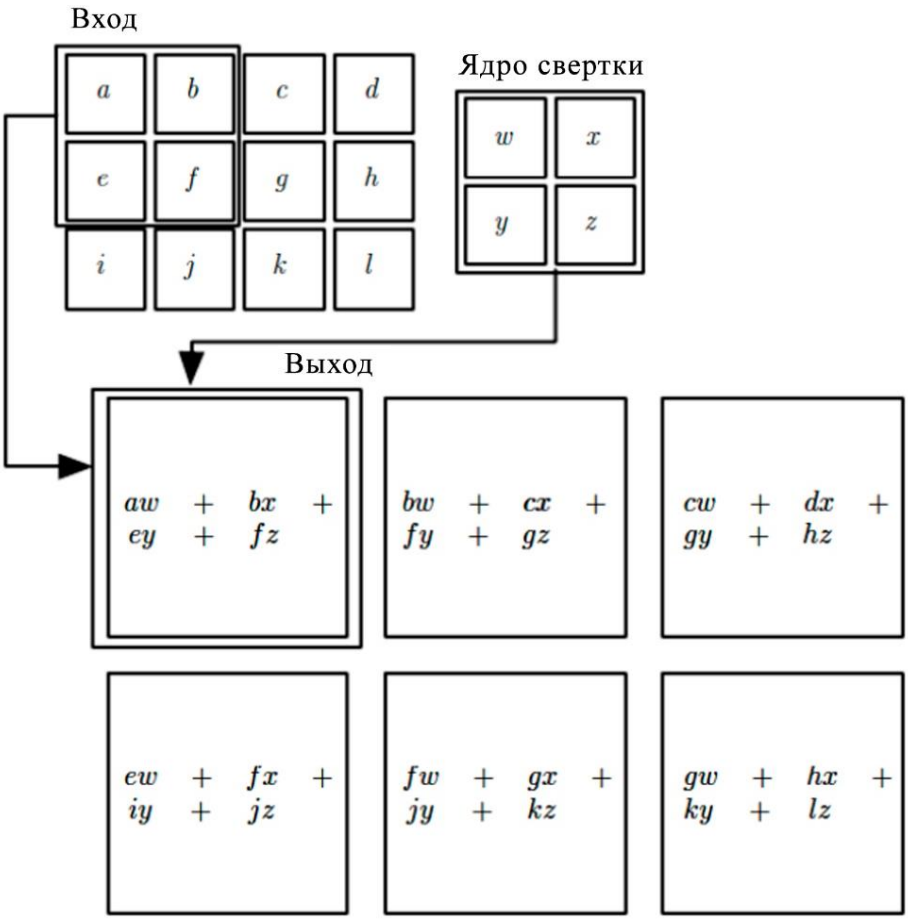

Рисунок 4 — Свертка одним фильтром

Главным назначением одного конкретного сверточного слоя является выделение простых шаблонов во входе с помощью обучаемых фильтров. Располагая сверточные слои один за другим и в комбинации с другими типами слоев, получаем, что с ростом глубины сети растет абстрактность и сложность выделяемых признаков. Так если фильтры настроены на выделение прямых линий на картинке, то после первого слоя будут распознаны только прямые линии, после второго — их комбинации, то есть, например, выпуклые фигуры и так далее;

2) Активационные слои.

Выход сверточного слоя есть линейной преобразование над его входом. Композиция линейных преобразований дает линейное преобразование. У нас же есть желание аппроксимировать функции высокой сложности с помощью нейронной сети. Поэтому необходимо добавлять нелинейности. Именно это и делает активационный слой. Обычно в качестве нелинейных функций в промежуточных слоях используют функцию $f(x) = max(0, x)$, называемую «выпрямитель» (ReLU) [6] и ее параметрические аналоги. Использование традиционных функций активации, таких как сигмоида или гиперболический тангенс, несет за собой проблемы размывающихся или взрывающихся градиентов на этапе обучения;

3) Слои подвыборки (субдискретизации).

В самом общем виде подвыборка — замена элементов входа на некоторую статистику его близлежащих элементов. Основная функция таких слоев — снижение размера представления данных после очередного слоя, для уменьшения числа параметров, количества вычислений в сети и избегания переобучения. Наиболее часто используемый вид статистики — максимум, когда оператор возвращает максимум из области, поданной на вход. Пример такой подвыборки изображен на рисунке 5. Также реже используются другие функции — среднее по области; взвешенное среднее с весами, пропорциональными расстоянию до центральной точки и другие. Выбор конкретной функции зависит от специфики задачи.

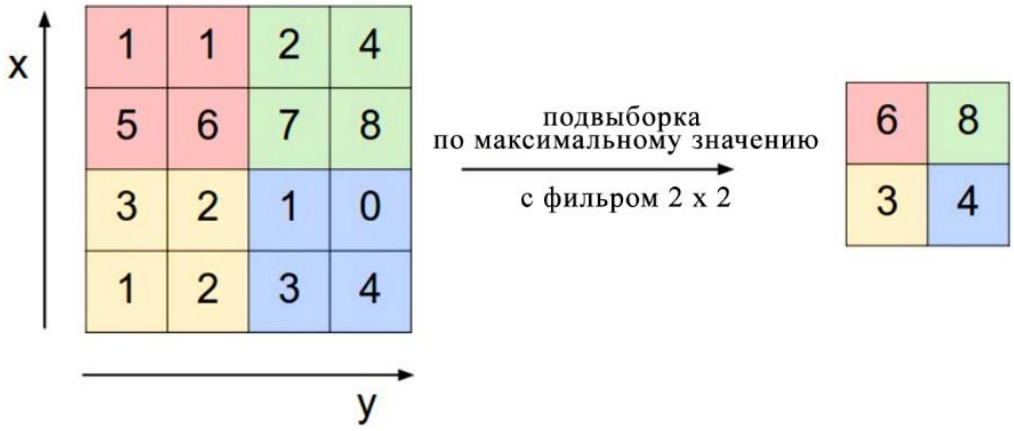

Рисунок 5 — Подвыборка по максимальному значению

Наряду со снижением размера данных, слои подвыборки делают представление данных приближенно инвариантным относительно малых трансляций признака во входных данных, так как при маленьком сдвиге во входных данных, значения на выходе слоя подвыборки почти не будут меняться, потому что происходит агрегация с помощью статистики по определенной окрестности. Это свойство может быть полезно, если мы больше интересуемся наличием признака во входе, нежели точным его положением. Чтобы сделать итоговое представление инвариантным не только к трансляциям, но и к другим преобразованиям, например, вращениям, можно агрегировать в слое подвыборки выходы из сверточных слоев параметризованных по-разному, т.е. имеющих различные формы и веса фильтров;

4) Полносвязные слои.

Все предыдущие слои, скомбинированные в некоторую структуру, по сути своей являются генератором нового представления входных данных. По полученным признакам, используя обыкновенную нейронную сеть из нескольких полносвязных слоев, производится классификация.

Отметим также, что наиболее часто используемая архитектура сверточных нейронных сетей выглядит следующим образом: нескольких сверточных слоев, следующих друг за другом, затем, как правило, нелинейная

функция активации и слой подвыборки. Такая последовательность может повторяться несколько раз.

На рисунке 6 изображен результат обучения сверточной нейронной сети, а точнее признаки на различных уровнях её глубины.

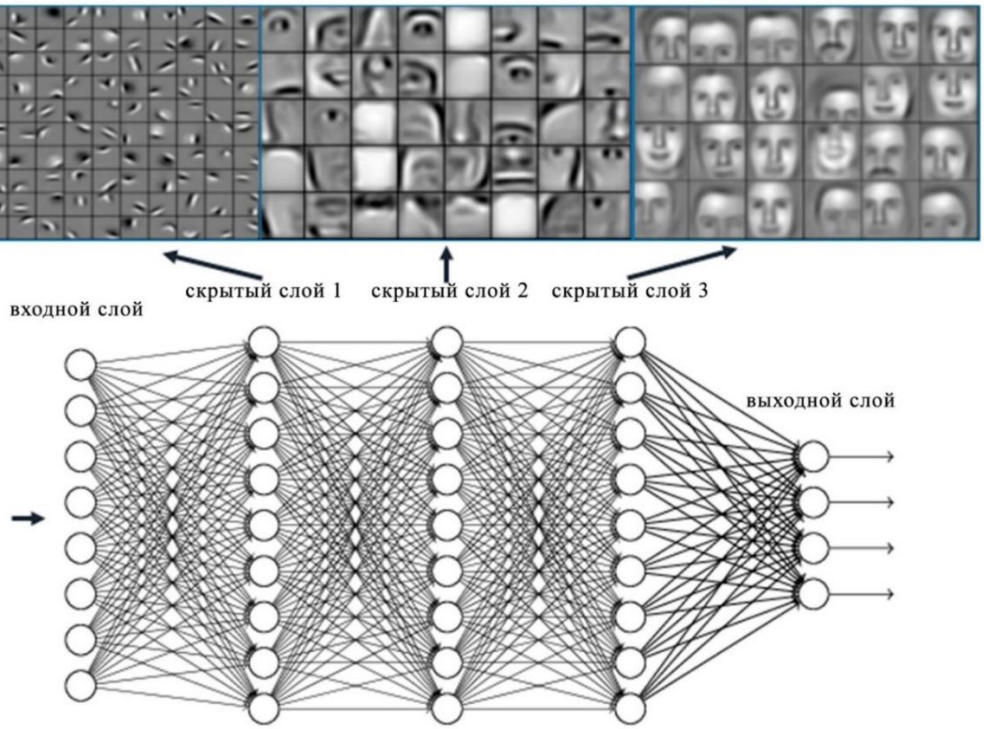

Рисунок 6 — Визуализация признаков на различных уровнях нейронной сети

1.5 Вывод по главе 1

В данной главе проведен аналитический обзор теоретических сведений о нейронных сетях. Были введены понятия искусственного нейрона, искусственной нейронной сети, рассмотрены архитектуры нейронных сетей и алгоритмы их обучения. Проведено исследование структуры сверточных нейронных сетей.

Для решения задачи процедурной генерации изображений была выбрана сверточная нейронная сеть, архитектура глубокого обучения, изначально нацеленная на распознавание изображений с целью их классификации.

2 Применение нейронных сетей в задачах обработки изображений

2.1 Глубокое обучение

На заре зарождения искусственного интеллекта, его развитие было направлено в сторону решения задач, которые были сложны для человеческого разума, но относительно просты и прямолинейны для компьютеров - задач, которые можно описать формальными математическими правилами. Настоящий вызов искусственному интеллекту — это решение тех задач, которые легко выполняются людьми, но трудны для описания формальным языком. Такие задачи человек решает интуитивно, автоматически — например: распознавание произнесенных слов или лиц на фотографии.

Глубокое обучение позволяет компьютерам учиться на опыте и понимать наш мир в терминах иерархии понятий, причем каждое понятие определяется через его отношение к более простым понятиям. Сбор знаний посредством опыта, позволяет избежать необходимости формального уточнения полученных знаний человеком. Иерархия понятий позволяет компьютеру изучать сложные понятия, выстраивая их и более простых. Если нарисовать граф, отражающий как понятия построены друг над другом, то он будет чрезвычайно глубоким. Именно поэтому, такой подход называется глубоким обучением [7].

Ранние успехи искусственного интеллекта в основном связаны с формальными средами и компьютерам не требовалось иметь большие знания о мире. К примеру, шахматный суперкомпьютер IBM Deep Blue, который победил чемпиона мира по шахматам Гарри Каспарова. Шахматы — довольно формализованный мир, содержащий всего 64 клетки и 32 фигуры, которые могут ходить только по строго определенным правилам. Несомненно, разработка успешной стратегии для игры в шахматы — грандиозное достижение, но разве вызов для искусственного интеллекта, описать набор фигур и допустимые ходы в компьютере. Проход шахматной доски с фигурами полностью описывается небольшим списком формальных правил.

Абстрактные и формальные задачи, являющиеся одними из самых сложных умственных задач для человека, относятся к числу самых простых для компьютера. Компьютеры уже давно способны победить даже лучшего шахматиста-человека, но только недавно получили способность среднестатистического человека для распознавания объектов и речи. Жизнь любого человека требует огромных знаний об окружающем мире. Значительная часть этих знаний субъективна и интуитивна, поэтому ее невозможно формализовать. Компьютеры, также должны иметь возможность фиксировать знания такого рода, чтобы вести себя разумно. Одна из ключевых проблем искусственного интеллекта — как получить эти неформальные знания [8].

В нескольких проектах по искусственному интеллекту разработчики стремились максимально точно кодировать знания о мире на формальных языках. Искусственный интеллект в таких проектах имел возможность автоматически формулировать утверждения, по заложенным формальным правилам, с использованием правил логического вывода. Такой подход называется подходом с базой знаний. Коммерческий успех таких систем не был достигнут. Одним из самых известных проектов является Cyc [8]. Cyc представлял собой базу знаний констант на языке CycL и механизм вывода. Сами константы вносились пользователем вручную, что представляет собой затруднительный процесс. На протяжении многих лет велись разработки в области создания формальных правил для точного описания мира. Например, Cyc не смог понять рассказ о человеке по имени Фрэд, который брился утром. Механизм вывода обнаружил непоследовательность в истории: он знал, что у людей нет электрических частей, но поскольку Фрэд держал электрическую бритву, он считал, что во время самого бритья Фрэд содержал в себе электрическую деталь. Поэтому он спросил, был ли Фрэд все еще человеком, когда он брился [8].

Сложности, с которыми сталкиваются системы, основанные на строго формализованных знаниях, свидетельствуют о том, что системы искусственного интеллекта должны обладать способностью приобретать

собственные знания, путем извлечения шаблонов из исходных данных. Эта способность более известна как машинное обучение. Внедрение машинного обучения позволило компьютерам решать задачи, связанные со знанием реального мира, и принимать решения, которые выглядят как субъективные. Алгоритм машинного обучения, называемый логистической регрессией, может, например, в системе медицинского назначения рекомендовать делать кесарево сечение, в зависимости от конкретных значений релевантных показателей. Алгоритм машинного обучения, называемый наивный Байесовский классификатор, может отделять полезную электронную почту от спама.

Производительность алгоритмов машинного обучения полностью зависит от представления данных, которые они производят [8]. Например, когда система с логистической регрессией используется для рекомендации кесарева сечения, она не рассматривает пациента напрямую. Вместо этого врач сообщает системе релевантную информацию, такую как наличие либо отсутствие рубца матки. Каждая часть информации, включенная в представление пациента, называется признаком. Система с логистической регрессией изучает, как каждый из этих признаков коррелирует с различными возможными исходами. Тем не менее, система изнутри никак не влияет на появление эти признаков. Если бы система получила магнитно-резонансную томографию пациента, а не формализованный отчет врача, то она не смогла бы сделать полезный прогноз.

Зависимость от представления — это общее явление, которое влияет на информационные технологии и даже на нашу повседневную жизнь. В информатике такие операции, как поиск набора данных, могут выполняться экспоненциально быстрее, если этот набор данных структурирован и индексирован. Люди легко выполняют арифметические операции с арабскими цифрами, но арифметика с римскими цифрами куда более трудоемкий процесс. Поэтому и не удивительно, что выбор представления имеет огромное влияние на производительность алгоритмов машинного обучения. Визуальная иллюстрация представлена на рисунке 7.

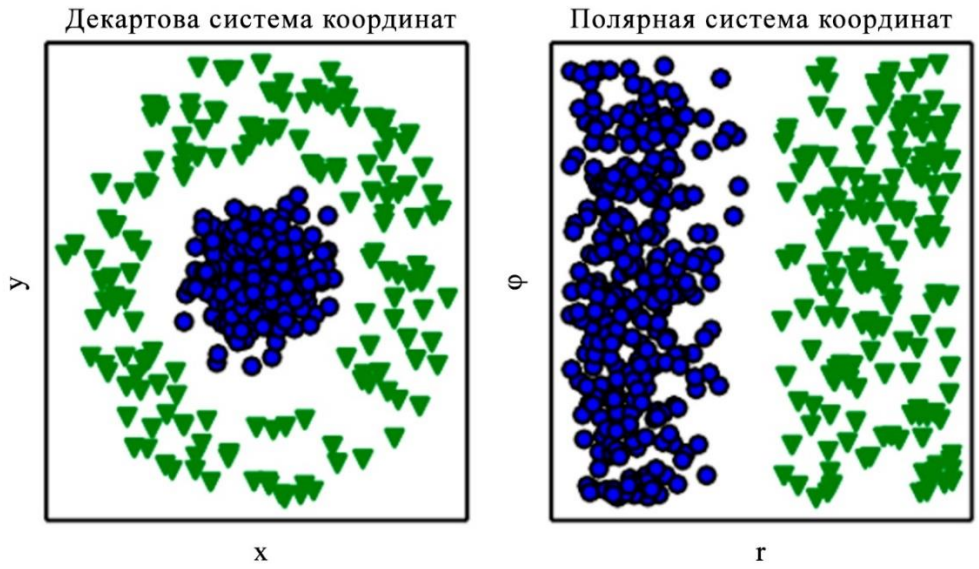

Рисунок 7 — Пример различного представления данных

Например, если стоит задача разделить две категории данных, проведя линию между ними на точечном графике разброса. Очевидно, что выполнить такую задачу на левом графике, с использованием декартовой системы координат, невозможно. Однако, на графике справа данные представлены в полярной системе координат, и задача становится простой для решения.

Многие задачи искусственного интеллекта могут быть решены путем разработки набора полезных признаков, конкретно подходящих для текущей задачи, и последующая обработка этих признаков алгоритмом машинного обучения [7]. К примеру, полезным признаком для распознавания речи является размер голосового тракта. Этот признак дает четкое представление о том, является ли говорящий мужчиной, женщиной или ребенком.

Однако, для большинства задач трудно понять какие признаки следует извлекать. Например, при необходимости решения задачи обнаружения автомобилей на фотографиях. Мы знаем, что у автомобилей есть колеса, поэтому мы можем использовать колесо как признак. Описать как точно выглядит колесо в терминах яркостей пикселей — сложная задача. Колесо имеет простую геометрическую форму, но изображение может быть зашумлено тенями, падающими на колесо, солнечными бликами, крылом автомобиля и так далее.

Одним из решений этой проблемы стало использование машинного обучения не только для отображения пути от формализованного представления к результату, но и представлению данных внутри системы. Этот подход называется обучение представлению. Полученные в результате обучения представления, приводят к более высокой производительности, чем представления, разработанные вручную. Также они позволяют системам искусственного интеллекта быстрее адаптироваться к новым задачам с минимальным вмешательством человека. Алгоритм обучения представлений может обнаружить сложный набор признаков для решения простой задачи за считанные минуты, либо сложной задачи за часы, вместо месяцев. Ручное проектирование признаков для сложной задачи может занять несколько лет для целой команды разработчиков.

Примером алгоритма обучения представлений является автокодировщик. Автокодировщик представляет собой комбинацию кодировщика, который преобразует входные данные в различные представления, и декодировщика, который преобразует эти представления обратно в исходный формат [9]. Задачей кодировщика является формирование сжатого представления входных данных. Декодировщик представляет собой отражение кодировщика и предназначен для восстановления исходных данных с максимально возможной точностью. В процессе обучения декодировщик стимулирует формирование наиболее информативного сжатого представления. Чем точнее восстановленный вход соответствует оригиналу, тем лучше сжатое представление. Различные виды автокодировщиков используются для получения различных признаков.

При разработке признаков или алгоритмов их обучения, цель состоит в том, чтобы разделить факторы вариации, которые объясняют наблюдаемые данные. Факторы, в данном случае, определяются как отдельные источники влияния. Такие факторы, чаще всего, не являются наблюдаемыми величинами. Они существуют как некие конструкции в человеческом разуме, которые обеспечивают упрощение понимания наблюдаемых данных. Их можно

рассматривать как концепции или абстракции, которые помогают нам понимать богатую изменчивость данных. Когда человек анализирует речь, факторы вариации включают в себя возраст говорящего, его пол, акцент и слова, которые он произносит. При анализе образа автомобиля факторы вариации включают в себя положение автомобиля, его цвет, а также угол падения света и яркость сцены.

Факторы вариации влияют на каждую отдельную часть данных, которые мы можем наблюдать. Яркость отдельных пикселей на изображении красной машины, может быть очень близка к черной в ночное время суток. Форма силуэта автомобиля зависит от угла обзора. Поэтому, большинство систем требует, чтобы факторы вариации фильтровались и отбрасывались менее важные.

Извлечение высокоуровневых абстрактных признаков является трудной задачей. Большинство факторов вариации, например, акцент, могут быть определены только с использованием сложного, практически человеческого уровня понимания информации.

Глубокое обучение решает проблему обучения представлений, путем внедрения таких представлений, которые выражаются через другие, более простые представления [7]. Глубокое обучение позволяет конструировать сложные понятия из простых. На рисунке 8, показано как система глубокого обучения представляет образ человека, комбинируя более простые понятия, такие как углы и контуры, которые в свою очередь определены через понятие ребер.

Компьютеру сложно понять смысл необработанных данных, представляющих собой совокупность пикселей. Функция, которая преобразовывает набор пикселей в идентификатор объекта, чрезвычайно сложна. Глубокое обучение позволяет решить эту задачу, путем разбиения сложного преобразования, на несколько простых преобразований, каждое из которых описывается своим слоем модели. Вход модели представляет собой видимый слой, потому что содержит переменные, которые мы можем

наблюдать. Далее следует серия скрытых слоев, извлекающих из изображения абстрактные признаки. Эти слои называются скрытыми, потому что их значения не указаны в данных изображения. Модель в автоматическом режиме определяет какие из понятий полезны для объяснения отношений в наблюдаемых данных. Изображения в окружностях представляют собой визуализацию признаков, представленных в каждом скрытом слое. Первый скрытый слой может легко определять ребра, сравнивая яркости соседних пикселей. Учитывая описание первого скрытого слоя, второй скрытый слой может находить углы и контуры, которые представляют собой коллекции ребер. Учитывая описание второго скрытого слоя, третий скрытый слой способен обнаруживать целые части конкретных объектов, собирая их из определенных наборов контуров и углов. В итоге, описание изображения в терминах, содержащихся в нем частей объектов, может использоваться для распознавания объектов, присутствующих на нем.

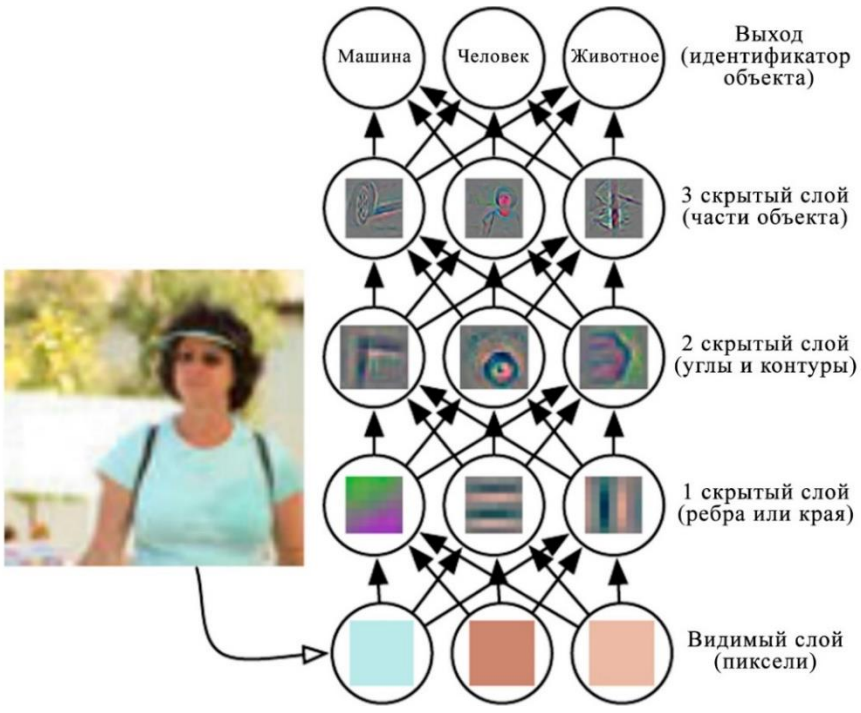

Рисунок 8 — Модель глубокого обучения

Таким образом, глубокое обучение — это один из подходов к решению задач искусственного интеллекта, позволяющий информационным системам

развиваться и улучшаться, получая опыт и знания. Глубокое обучение — это особый вид машинного обучения, который достигает большой мощности и гибкости, изучая мир, как вложенную иерархию понятий, где каждое понятие определяется отношением к более простым понятиям, и более абстрактными представлениями, вычисляемыми в терминах менее абстрактных представлений.

2.2 Применение нейронных сетей

В последние годы наблюдается повышенный интерес к нейронным сетям, которые нашли применение в самых различных областях человеческой деятельности - бизнесе, медицине, технике. Нейронные сети используются при решении задач прогнозирования, классификация, управления. Такой впечатляющий успех определяется несколькими причинами [10].

Широкий круг задач, решаемый с помощью нейронных сетей, не позволяет в настоящее время создавать универсальные, мощные сети, вынуждая разрабатывать специализированные нейронные сети, функционирующие по различным алгоритмам. Модели нейронных сетей могут быть программного и аппаратного исполнения.

Тем не менее, искусственные нейронные сети основаны на весьма простой биологической модели нервной системы.

Искусственные нейронные сети нашли применение во многих областях техники, где они используются для решения многочисленных прикладных задач:

- Распознавание образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.

- Кластеризация/категоризация. При решении задачи кластеризации, которая известна также как классификация образов "без учителя", отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

- Аппроксимация функций. Предположим, что имеется обучающая выборка$((x_1, y_1), (x_2, y_2) \dots, (x_n, y_n))$(пары данных вход-выход), которая генерируется неизвестной функцией (x), искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции (x). Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

- Предсказание/прогноз. Пусть заданы n дискретных отсчетов $\{y(t_1),\ y(t_2) \dots, y(t_n)\}$ в последовательные моменты времени $t_1, t_2 \dots, t_n$. Задача состоит в предсказании значения $y(t_{n+1})$ в некоторый будущий момент времени t_{n+1}. Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

- Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, относящаяся к классу NP-полных, является классическим примером задачи оптимизации.

- Память, адресуемая по содержанию. В модели вычислений Фон-Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержанию, доступна по

указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

- Управление. Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ является входным управляющим воздействием, а $y(t)$ - выходом системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия $u(t)$, при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

2.3 Распознавание изображений

Обработка изображений с целью их распознавания является одной из центральных и практически важных задач при создании систем искусственного интеллекта. Проблема носит явно выраженный комплексный иерархический характер и включает ряд основных этапов: восприятие поля зрения, предобработка, сегментация, нормализация выделенных объектов, распознавание. Такой важный обязательный этап как понимание (интерпретация) изображений включается частично в этап сегментации и окончательно решается на этапе распознавания [11].

Основным элементом любой задачи распознавания изображений является ответ на вопрос: относятся ли данные (входные) изображения к классу изображений, который представляет данный эталон? Казалось бы, ответ можно получить, сравнивая непосредственно изображение с эталонами (или их признаки).

Для решения задачи в целом и на отдельных ее этапах применяются различные методы сегментации, нормализации и распознавания. Литературные источники и многолетний опыт работы в области обработки зрительных картин

позволяет предложить классификацию методов обработки и распознавания изображений в соответствии со схемой, приведенной на Рисунке 9 [12, 13, 14].

Рисунок 9 — Схема этапов распознавания изображения

Методы распознавания образов делятся на четыре основные категории:

- методы, основанные на теории решений;
- методы, основанные на ортогональных преобразованиях;
- структурные методы;
- методы, основанные на теории нейронных сетей.

В распознавании образов центральную роль играет принцип «обучения» на выборке известных образов [15].

Первая категория имеет дело с образами, описанными с помощью количественных дескрипторов, таких как длинна, площадь, текстура. Признаком изображения называется его простейшая отличительная характеристика или свойство. Некоторые признаки являются естественными в том смысле, что они устанавливаются визуальным анализом изображения, тогда как другие, так называемые искусственные признаки. Получаются в результате его специальной обработки или измерений. К естественным признакам относятся яркость и текстура различных областей изображения, форма контуров объектов.

В этой категории дескрипторы представлены в виде вектора признаков.

Рисунок 10 демонстрирует пример построения вектора признаков. В этом примере выбирается сигнатура в качестве способа представления объекта. Получаются одномерные сигналы. Каждая сигнатура может описываться как набор значений её амплитуды, для чего с заданным шагом проводится дискретизация по углу θ, которая дает последовательность точек отсчета $\theta_1, \theta_2 \dots, \theta_n$. Тогда вектор признаков можно формировать, присваивая его компонентам значения $x_1 = r(\theta_1), x_2 = r(\theta_2) \dots, x_n = r(\theta_n)$. Этим векторам соответствуют точки в n-мерном евклидовом пространстве, а класс образов представить себе в виде n-мерного «облака» в этом пространстве. В конкретной задаче распознавания изображения степень разделимости классов сильно зависит от выбора дескрипторов [16].

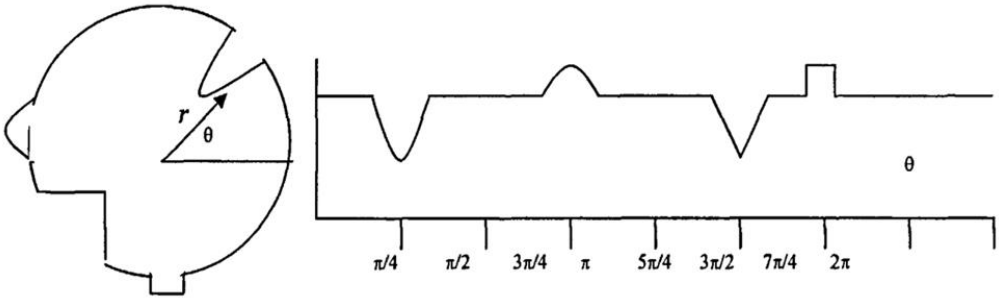

Рисунок 10 — Контур изображения и соответствующая сигнатура

В методах распознавания, основанных на сопоставлении, каждый класс представляется вектором признаков образа, являющегося прототипом этого класса. Незнакомы образ приписывается к тому классу, прототип которого оказывается ближайшим в смысле заранее заданной метрики. Простейший подход состоит в использовании классификатора, основанного на минимальном расстоянии, который, как ясно из названия, вычисляет, например, евклидовы расстояния между вектором признаков неизвестного объекта и каждым вектором прототипа. Решение о принадлежности объекта к определенному классу принимается по наименьшему из таких расстояний [17].

Вторая категория методов основана на ортогональных преобразованиях. Многие ортогональные преобразования практически используются в области

распознавания изображений. Например, преобразование Фурье, преобразование Адамара-Уолша и преобразование Карунена-Лоэва.

Третья категория ориентирована на образы, для описания которых лучше подходят качественные дескрипторы, например, реляционные [15]. Эта идея получила развитие в связи с реляционными дескрипторами. Они с равным успехом могут применяться для границ и для областей, и главная цель таких дескрипторов — зафиксировать в форме правил подстановки элементные конфигурации, которые повторяются на границе или внутри области. На рисунке 11 показана простая структура в виде лестницы. Предположим, что в результате сегментации на изображении была выделена такая структура. Определив два непроизводных элемента a и b, как показано на рисунке 11, можно закодировать фигуру в форме, показанной на рисунке 11 (в).

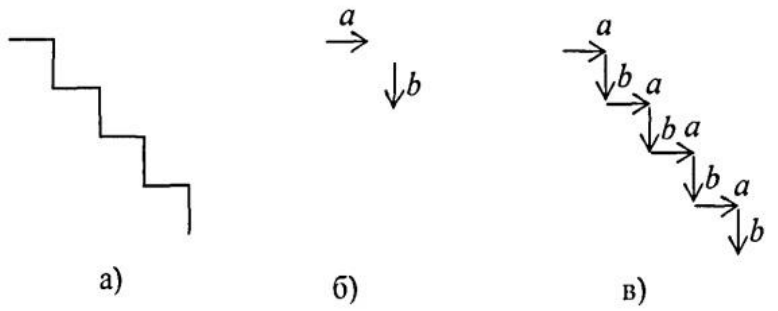

Рисунок 11 — а) Граница изображения; б) примитивные элементы; в) структура в закодированном виде

Более содержательным способом описания было бы определение примитивов a и b и формирование образа на их основе в виде символьной строки $w = abababab$.

Существуют прикладные задачи, в которых распознавание с простыми примитивными элементами не могут работать. Для распознавания сложных типов границ изображений необходимо создать разные примитивные элементы, например, типы линий [18]. Тогда сегменты границ объекта на изображении можно представить функциями одной переменной.

Четвертая категория имеет дело с нейронными сетями, которые могут использоваться на этапе выделения признаков и на этапе распознавания этих признаков или самых выделенных областей изображения [19]. В области распознавания изображений используются многослойные нейронные сети без обратной связи, нейронные сети Хопфилда и Хэмминга и персептроны [20]. Нейронные сети более устойчивые, чем другие статистические методы при распознавании изображений, если изображение на входе зашумлено.

2.3 Применение нейронных сетей к задаче генерации стилизованных изображений

Нейронные сети лежат в основе большинства современных систем распознавания и синтеза речи, а также распознавания и обработки изображений. Они применяются в некоторых системах навигации, будь то промышленные роботы или беспилотные автомобили. Алгоритмы на основе нейронных сетей защищают информационные системы от атак злоумышленников и помогают выявлять незаконный контент в сети.

До недавнего времени скорость работы нейронных сетей была слишком низкой, чтобы они могли получить широкое распространение, и поэтому такие системы в основном использовались в разработках, связанных с компьютерным зрением, а в остальных областях применялись другие алгоритмы машинного обучения.

Трудоёмкая и длительная часть процесса разработки нейронной сети — её обучение. Для того, чтобы нейронная сеть могла корректно решать поставленные задачи, требуется прогнать её работу на десятках миллионов наборов входных данных. Именно с появлением различных технологий ускоренного обучения и связывают распространение нейронных сетей.

Главным следствием популярности использования нейронных сетей можно считать появление технологий, которые позволяют делать нейронные сети, значительно меньше подверженные переобучению.

Во-первых, появился большой и общедоступный массив размеченных картинок (ImageNet), на которых можно обучаться. Во-вторых, современные видеокарты позволяют в сотни раз быстрее обучать нейросети и их использовать. В-третьих, появились готовые, предобученные нейронные сети, распознающие образы, на основании которых можно делать свои приложения, не занимаясь длительной подготовкой нейронной сети к работе. Всё это обеспечивает очень мощное развитие нейронных сетей именно в области распознавания образов.

В последние несколько лет появилось сразу несколько громких развлекательных проектов, использующих нейронные сети для генерации стилизованных изображений. Примерами систем процедурной обработки изображений являются онлайн сервисы: Ostagram, Deepart; мобильные приложения российских разработчиков: Prisma и Mlvch.

2.3.1 Онлайн сервис Ostagram

Онлайн сервис Ostagram [21] предназначен для обработки изображений. Сервис предоставляет возможность переноса стиля знаменитых художников на другие фотографии.

У сервиса есть бесплатная версия: она создаёт изображение в минимальном разрешении до 600 пикселей по наиболее длинной стороне картинки. Пользователь получает результат только одной из итераций наложения фильтра на фотографию.

Платных версий две: «Премиум» выдаёт картинку до 700 пикселей по самой длинной стороне и применяет к изображению 600 итераций обработки нейронной сетью (чем больше итераций, тем интереснее и интенсивнее обработка). Стоимость одного снимка составляет 50 рублей.

В версии «HD» можно настраивать количество итераций: 100 обойдутся в 50 рублей, а 1000 — в 250 рублей. При этом изображение будет иметь разрешение до 1200 пикселей по самой длинной стороне, и его можно будет

использовать для печати на холсте: Ostagram предлагает такую услугу с доставкой от 1800 рублей.

Для создания картин используются сверточные нейронные сети и алгоритм художественно стиля Леона Гатуса.

Пример работы сервиса Ostagram показан на рисунке 12.

Рисунок 12 — Пример работы Ostagram

2.3.2 Онлайн сервис Deepart

Сервис Deepart [22] применяется для создания арта на основе фотографий и художественных изображений, аналогично Ostagram.

Бесплатно можно сделать только картинку 500 × 500 пикселей, кроме того, в правой нижней части будет размещен логотип Deepart. Качественное изображение 1300 × 1300 пикселей и без водяных знаков обойдется в 19 евро.

Ожидание в очереди на получение результата занимает в Deepart значительное время — сутки и даже более. Такое длительное ожидание разработчики объясняют количеством желающих стилизовать свои фото.

Пример работы Deepart показан на рисунке 13.

Рисунок 13 — Пример работы Deepart

2.3.3 Мобильное приложение Prisma

Приложение Prisma [23] — это фоторедактор, который сочетает в себе технологии нейронных сетей и искусственного интеллекта. Он способен перерисовать любое изображение на манеру классиков: Мунка, Левитана, Пикассо или Ван Гога.

Главное преимущество Prisma по сравнению с аналогичными сервисами (Deepart и Ostagram) — скорость обработки изображений. Снимок загружается в облако и обрабатывается с помощью специальных алгоритмов, после чего результат загружается обратно на устройство. Процедура в среднем занимает 15 секунд — как утверждает генеральный директор Prisma Labs, это лучший показатель среди конкурентов.

В приложении доступно ограниченное количество стилей изображения, а также отсутствует возможность загружать свои изображения, для использования в качестве стиля.

Пример работы приложения Prisma показан на рисунке 14.

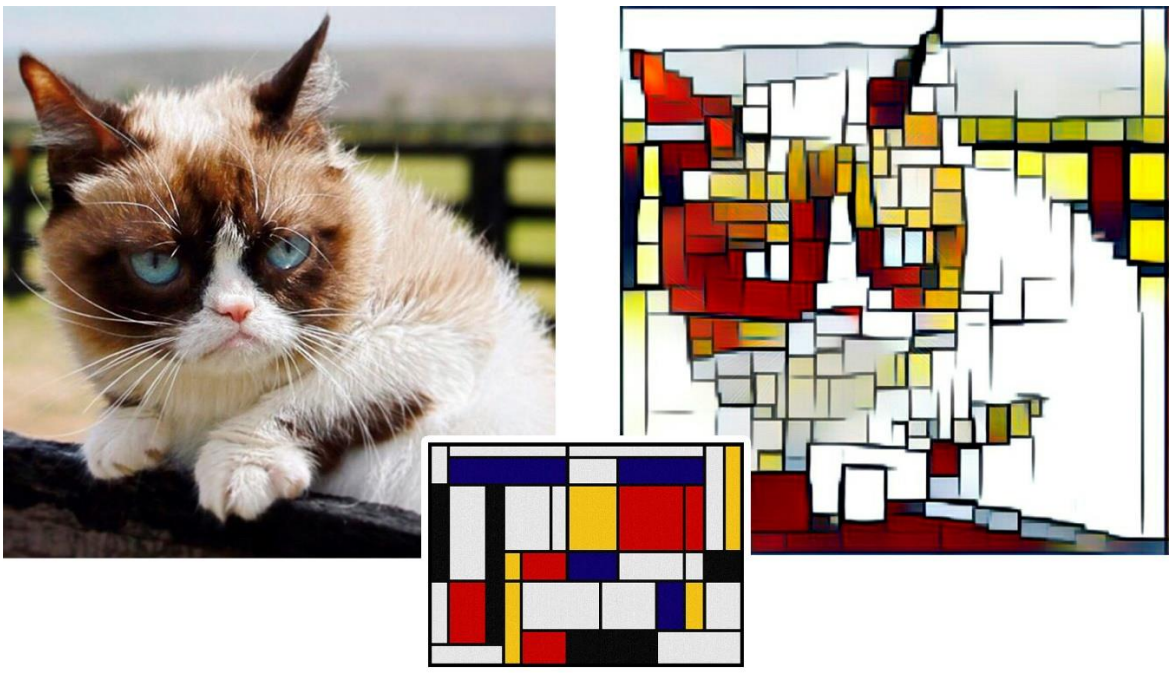

Рисунок 14 — Пример работы Prisma

2.3.4 Мобильное приложение Mlvch

Мобильное приложение Mlvch [24] работает по тому же принципу, что и Prisma, но отличается более детальной проработкой фотографий и большим количеством доступных стилей — в приложении Mlvch их 50. В приложении Prisma изображение обрабатывается за 20-30 итераций, а в приложении Mlvch — за 100, в итоге в мобильном приложении Mlvch результаты меньше похожи на оригинальную фотографию и больше — на художественное произведение: приложение передает не только стиль картины, выбранной в качестве фильтра, но и ее цветовую гамму и даже отдельные графические элементы.

Из минусов по сравнению с приложением Prisma у приложения Mlvch сравнительно низкая скорость обработки и высокая цена — бесплатно можно обработать только 5 фотографий, за каждую следующую придется платить по 75 рублей или можно приобрести пакет для 100 фотографий за 1590 рублей.

Процедура обработки изображения в мобильном приложении Mlvch в среднем занимает 2 часа.

Пример работы мобильного приложения Mlvch показан на рисунке 15.

Рисунок 15 — Пример работы мобильного приложения Mlvch

2.4 Вывод по главе 2

В данной главе рассмотрены области применения искусственных нейронных сетей глубокого обучения в задачах обработки изображений, описаны основные этапы и методы распознавания изображений. Представлены основные подходы к решению задачи генерации стилизованных изображений. Приведены примеры систем процедурной обработки изображений на основе искусственных нейронных сетей.

3 Разработка информационной системы процедурной обработки изображений искусственной нейронной сетью

3.1 Постановка задачи генерации изображения

Пусть $\vec{x}$ — генерируемое изображение для синтеза, $\vec{p}$ — изображение-содержимое и $\vec{a}$ — изображение-стиль. Изображение $\vec{x}$ представляет собой случайно сгенерированное изображение белого шума. L — множество слоев

сверточной нейронной сети. Каждый слой $l \in L$ характеризуется конечным числом фильтров и размером.

Пусть F — множество ответов фильтров (представление содержимого изображения), а G — множество корреляций между ответами фильтров (представление стиля изображения).

Для того чтобы реконструировать $\vec{p}$ и $\vec{a}$ в $\vec{x}$ необходимо минимизировать разницу между $F^l(\vec{p})$ и $F^l(\vec{x})$ (отклонение по содержимому), а также $G^l(\vec{a})$ и $G^l(\vec{x})$ (отклонение по стилю).

Соответственно, задача состоит в минимизации общей функции отклонения $\mathcal{L}(\vec{p}, \vec{a}, \vec{x})$.

3.2 Представление содержимого изображения

В процессе обработки изображения сверточная нейронная сеть обучается распознавать объекты, развивая представление изображения, которое делает информацию об объекте более явной и точной. Таким образом, в процессе обработки нейронной сетью входное изображение преобразуется в набор представлений, который отражает содержимое изображения более подробно, по сравнению с конкретными значениями пикселей. Следовательно, возникает возможность визуализировать информацию о входном изображении, содержащуюся в каждом слое нейронной сети, путем реконструкции изображения из карт признаков любого слоя [25]. Пример реконструкции на рисунке 16. Более высокие уровни сети захватывают высокоуровневое содержимое, с точки зрения объектов и их расположения на входном изображении, но при реконструкции не воспроизводят изображение с точностью до пикселя. Напротив, реконструкция из нижних слоев представляет собой воспроизведение изображения точно по пикселям, что отражено на рисунке 16. Поэтому, карты признаков из слоев высокого уровня можно считать представлением содержимого.

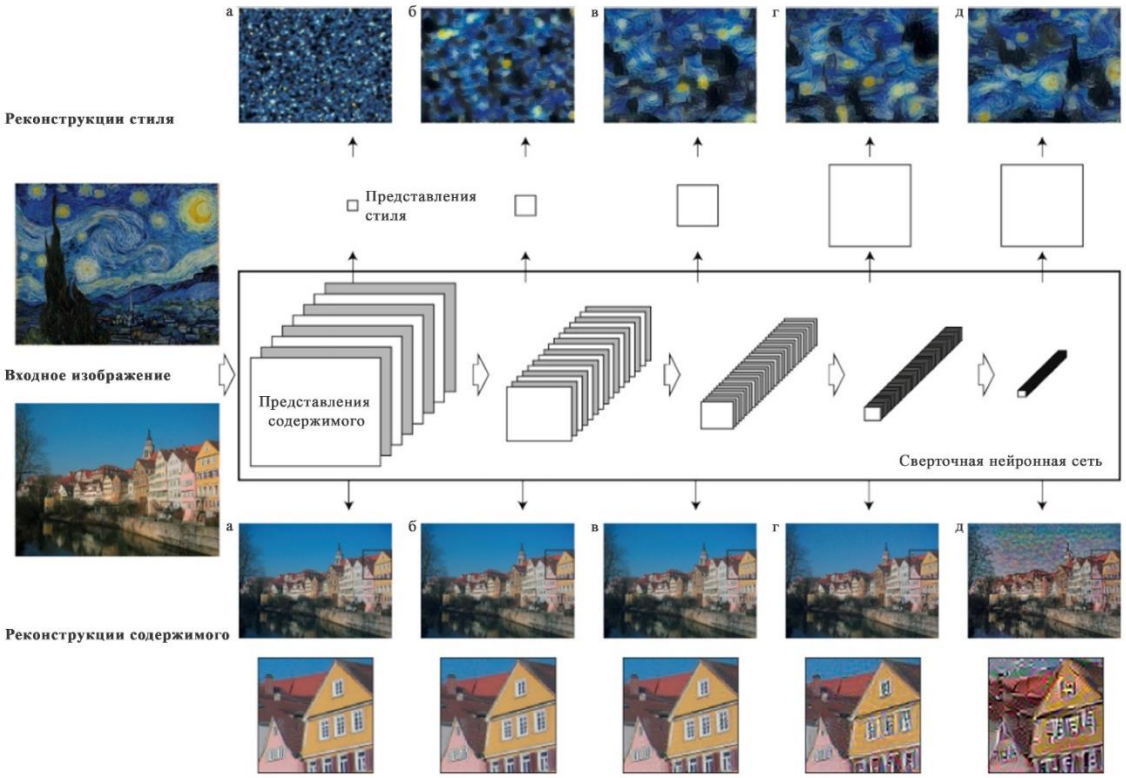

Рисунок 16 — Сверточная нейронная сеть. Реконструкция содержимого. Реконструкция стиля.

Каждый слой нейронной сети представляет собой набор нелинейных фильтров, сложность которых возрастает в зависимости от положения слоя в сети. Следовательно, изображение $\vec{x}$ кодируется в каждом слое сверточной нейронной сети посредством ответов фильтра на это изображение. Слой с N_l различными фильтрами имеет N_l карт признаков, каждая из которых размером M_l, где M_l это высота, умноженная на ширину карты признаков. Соответственно ответы слоя l можно хранить в матрице

$$F^l \in R^{N_l \times M_l}, \tag{8}$$

где F_{ij}^l — активация i-ого фильтра на позиции j в слое l.

Для визуализации информации об изображении, закодированной на разных слоях нейронной сети, мы выполняем градиентный спуск по изображению белого шума, чтобы найти другое изображение, которое

совпадает по ответам фильтров с признаками исходного изображения. Пусть $\vec{p}$ и $\vec{x}$ оригинальное и генерируемое изображение. P^l и F^l их соответствующее представление в слое l. Затем мы определяем среднеквадратическое отклонение между двумя представлениями признаков [25]:

$$\mathcal{L}_{\text{содержимого}}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^l - P_{ij}^l\right)^2. \quad (9)$$

Производная функции отклонения по активациям в слое l равна

$$\frac{\partial \mathcal{L}_{\text{содержимого}}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij} & \text{если } F_{ij}^l > 0 \\ 0 & \text{если } F_{ij}^l < 0 \end{cases} \quad (10)$$

Из этого следует что градиент по отношению к изображению $\vec{x}$, может быть вычислен с использованием стандартного метода обратного распространения ошибки [8]. Таким образом, происходит изменение первоначального случайного изображения $\vec{x}$, до тех пор пока оно не генерирует тот же ответ в определенных слоях сверточной нейронной сети. Реконструкции содержимого на рисунке 16 основаны на слоях сети VGG-19 [25].

3.3 Представление стиля изображения

Чтобы получить представление стиля входного изображения, используется пространство признаков, изначально предназначенное для захвата текстуры. Это пространство признаков построено поверх ответов фильтров в каждом слое нейронной сети. Оно представляет собой корреляции между различными ответами фильтров по пространственной протяженности карт признаков. Включая корреляции признаков нескольких слоев, мы получаем многомасштабное представление входного изображения, которое захватывает текстурную информацию, без привязки к расположению.

Таким образом, возможно визуализировать информацию при помощи пространства признаков стиля, взятых с различных уровней нейронной сети, путем реконструкции изображения, которое будет соответствовать представлению стиля входного изображения [26]. В действительности, реконструкция изображения по признакам стиля производит текстурированную версию входного изображения, которая сохраняет его общий вид, с точки зрения цвета и локальных структур. Более того, размер и сложность локальных структур входного изображения возрастает по иерархии нейронной сети, что связано с увеличением размеров рецептивных полей и сложности объектов. Из чего можно заключить, что данное многомасштабное представление изображения можно считать представлением стиля.

Как сказано выше, представление стиля являет собой многомасштабное представление изображения, которое включает в себя несколько слоев нейронной сети. Так же, стиль можно задать более локально, используя меньшее число нижних слоев, что в итоге приведет к ощутимым визуальным различиям. При использовании слоев представления стиля высоких уровней, локальные структуры изображения совпадают в более широких масштабах, что приводит к более плавному и непрерывному визуальному отображению. Таким образом, наиболее привлекательные изображения в визуальном плане, обычно создаются путем сопоставления представлений стиля высоких уровней нейронной сети, что отчетливо видно в последней колонке рисунка 17.

В каждом слое сверточной нейронной сети было построено представление стиля, которое вычисляет корреляции между различными ответами фильтра, где ожидание берется по пространственной протяженности входного изображения. Эти корреляции признаков задаются матрицей Грамма

$$G^l \in R^{N_l \times N_l}, \tag{11}$$

где G_{ij}^l — скалярное произведение между векторными картами признаков i и j в слое l [27]:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}. \quad (12)$$

Рисунок 17 — Результат соответствия представлений стиля и содержимого относительно глубины выбранных слоев

Чтобы создать текстуру, соответствующую стилю изображения мы используем градиентный спуск по изображению белого шума, чтобы найти другое изображение, которое соответствует представлению стиля исходного изображения. Это делается путем минимизации среднеквадратичного расстояния между элементами матрицы Грамма исходного изображения и элементами матрица Грамма генерируемого изображения. Итак, пусть $\vec{a}$ и $\vec{x}$ оригинальное изображение и генерируемое изображение. A^{l} и G^{l} их соответствующее представление стиля в слое l. Влияние слоя l на общее отклонение выражается [27]

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} \left(G_{ij}^l - A_{ij}^l\right)^2, \tag{13}$$

тогда общее отклонение

$$\mathcal{L}_{\text{стиля}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l. \tag{14}$$

где w_l — весовой коэффициент вклада каждого слоя в глобальное отклонение.

Производную E_l по активациям в слое l можно вычислить аналитически:

$$\frac{\partial \mathcal{L}_{\text{стиля}}}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left((F^l)^T (G^l - A^l)\right)_{ij} & \text{если } F_{ij}^l > 0 \\ 0 & \text{если } F_{ij}^l < 0 \end{cases} \tag{15}$$

Соответственно, градиент E_l по отношению к активациям, может быть вычислен с использованием стандартного метода обратного распространения ошибки [4]. Реконструкций стиля на рисунке 16 сгенерированы на основе представлений стиля в слоях сети VGG-19: «conv1_1» (а), «conv1_1» и «conv2_1» (б), «conv1_1», «conv2_1» и «conv3_1» (в), «conv1_1», «conv2_1», «conv3_1» и «conv4_1» (г), «conv1_1», «conv2_1», «conv3_1», «conv4_1» и «conv5_1» (д).

3.4 Синтез изображений

Ключевым моментом работы является то, что представление содержимого и представление стиля в сверточной нейронной сети между собой разделены. То есть, мы можем управлять обоими представлениями независимо для того, чтобы генерировать новые изображения, визуально похожие на исходные. Для демонстрации этого вывода, мы генерируем изображения, которые смешивают представление содержимого и представление стиля из двух разных исходных изображений.

Изображения синтезируются путем поиска изображения, которое одновременно соответствует представлению содержимого изображения и представлению стиля соответствующего произведения искусства. На синтезированном изображении сохраняется глобальное расположение исходного изображения, в тоже время цвета и локальные структуры заимствуются из произведения искусства. Внешний вид синтезированного изображения напоминает произведение искусства, даже если в качестве содержимого использована обыкновенная фотография.

Конечно, стиль и содержимое изображения невозможно перемешать полностью. При синтезе изображения, которое объединяет содержимое одного изображения со стилем другого, обычно не существует идеального изображения, соответствующего ограничениям обоих. Однако, функция потерь, минимизируемая при синтезе изображений содержит два термина для содержимого и для стиля. Поэтому возможно плавно регулировать акцент либо на воссоздании стиля, либо содержимого. При акценте на стиле, получаются изображения, представляющие собой текстурированную версию, лишенную содержания, в соответствии с первой колонкой рисунка 8. При акцентировании внимания на содержимом можно четко идентифицировать изображение, но при этом стиль картины не будет подобран достаточно узнаваемо. Соответственно, для каждой пары исходных изображений возможно найти такой компромисс между стилем и содержимым, чтобы синтезированное изображение максимально удовлетворяло ожидания.

Проблему синтезирования изображения в стиле известного произведения искусства, обычно рассматривают в области машинного зрения, так называемый нефотореалистичный рендеринг [28]. Предыдущие подходы в основном опирались на непараметрические методы для прямого управления пиксельным представлением изображения. В отличии от этих методов, использование глубоких нейронных сетей, обученных распознаванию изображений, позволяет проводить манипуляции в пространстве признаков, которые явно представляют высокоуровневое содержимое изображения.

Функции глубоких нейронных сетей, обученных распознаванию объектов, ранее использовались для распознавания изображения с целью классификации произведений искусства в соответствии с периодом их создания. В них классификаторы проходили обучение поверх сырых активаций нейронной сети, которые назывались представлением содержимого. Вероятнее всего, преобразование в пространство неподвижных объектов, таких как представления стиля, может обеспечить более высокую производительность при решении задач классификации изображений.

Метод синтеза изображений, который смешивает содержимое и стиль и разных исходных изображений, представляет собой относительно новый, увлекательный инструмент для изучения и восприятия нейронного представления внешнего вида искусства, стиля и содержания в целом. Синтез изображений может быть использован в широком спектре экспериментальных исследований в области визуального восприятия, от психофизики до функциональной визуализации и даже электрофизиологических нейронных записей. Нейронные представления могут независимо фиксировать содержание изображения и стиль, в котором оно представлено. Математическая форма представления стиля порождает ясную, проверяемую гипотезу о представлении изображения до уровня одного единственного нейрона. Представления стиля вычисляют корреляции между различными типами нейронов в сети. В свою очередь, извлечение корреляции между нейронами является биологически вероятным вычислением, которое, например, реализуется так называемыми комплексными клетками головного мозга в первичной зрительной системе.

Система нейронов головного мозга, которая обучена выполнять одну из основных вычислительных задач биологического зрения, способна автоматически изучать представления изображений, которые и отделяют содержимое от стиля. Объяснение может заключаться в том, что нейронная сеть головного мозга при распознавании объектов должна быть инвариантной для всех вариаций изображения, которые однозначно идентифицируют объект. Таким образом, человеческая способность абстрагировать содержание от стиля

дает возможность создавать и наслаждаться искусством, что является выдающимся результатом огромной мощности вывода из нашей с вами зрительной системы.

Чтобы сгенерировать изображение, которое смешивает содержимое изображения и стиль картины, в работе минимизируется расстояние между белым шумом и представлением содержимого изображения в одном из слоев сверточной сети, а также расстояние между белым шумом и представлениями стиля из нескольких слоев сверточной нейронной сети [26]. Пусть $\vec{a}$ и $\vec{p}$ художественное произведение и исходное изображение. Тогда, минимизируемая функция отклонения будет иметь вид [27]

$$\mathcal{L}_{\text{глобальная}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{содержимого}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{стиля}}(\vec{a}, \vec{x}) \quad (16)$$

где α и β — весовые коэффициенты для содержимого и для стиля.

На рисунке 17 отражены результаты генерации изображения с различными весовыми коэффициентами для содержимого и для стиля. Для представления стиля использовались следующие слои: «conv1_1» (А), «conv1_1» и «conv2_1» (Б), «conv1_1», «conv2_1» и «conv3_1» (В), «conv1_1», «conv2_1», «conv3_1» и «conv4_1» (Г), «conv1_1», «conv2_1», «conv3_1», «conv4_1» и «conv5_1» (Д). Коэффициент w_l определялся как единица, деленная на количество активных слоев с не нулевым значением w_l.

3.5 Используемые технологии разработки

Для решения поставленной задачи использовались следующие технологии и средства:

- скриптовый язык программирования Lua;
- библиотека Torch;
- фреймворк Caffe;
- модель VGG-19.

3.5.1 Скриптовый язык программирования Lua

Lua (от португальского “луна”) — облегченный скриптовый язык с расширяемой семантикой [29]. Lua был создан и поддерживается представителями Папского католического университета Рио-де-Жанейро. У него нет официального стандарта, и стандартом считается описание в руководстве пользователя.

В настоящее время Lua используется в индустрии игр, а также в ряде приложений в других предметных областях. Lua является сравнительно новым языком и позаимствовал черты и идеи из ряда более старых языков:

- синтаксис структур управления логикой программы — из языка программирования Modula;
- семантику более поздних версий — из функционального языка программирования Scheme;
- концепцию локальных переменных — из языка программирования C++;
- концепцию наличия единственной встроенной структуры данных, используемой несколькими способами — из языка программирования Lisp;
- использование ассоциативных массивов — из языка программирования SNOBOL;
- множественные присвоения и возвраты из функций — из языка программирования CLU.

Основополагающим принципом Lua является расширяемость семантики, т. е. предоставление мета-механизмов для реализации переменного набора инструментов вместо предоставления фиксированного набора инструментов. Это позволяет языку быть небольшим и простым, в то же время сохраняя мощность.

Lua поддерживает логические, числовые и строковые атомарные типы данных. Единственным уникальным типом данных является таблица — гетерогенный ассоциативный массив, позволяющий использовать разные типы

данных для разных пар ключей и значений. Функции являются объектами первого класса, т. е. ими можно манипулировать точно так же, как переменными, передавать и получать как аргументы и так далее.

Эти два основные свойства позволяют реализовывать многие структуры данных и принципы, доступные в других языках, при помощи использования разных типов данных для ключей и значений ассоциативного массива:

- структуры: используем строки (имена полей) как ключи и атомы любых нужных типов (значения полей) как значения; для этого случая Lua предоставляет специальный синтаксис, позволяющий обращаться к значениям полей по имени поля;
- массивы: используем целые числа (индексы) как ключи и атомы одного нужного типа (элементы массива) как значения;
- множества: элементы множества можно хранить либо как ключи, либо как значения;
- ассоциативные массивы;
- пространства имен: таблица может использоваться для хранения функций и переменных, относящихся к определенной предметной области;
- прототипы: Lua поддерживает объектно-ориентированную парадигму, позволяя хранить функции и данные, описывающие один объект, в одной таблице.

Таким образом, Lua компилируется в байт-код, исполняемый на виртуальной машине Lua. Lua — скриптовый язык, созданный для встраивания в другие языки, поэтому он предоставляется с интерфейсом прикладного программирования (API).

3.5.2 Библиотека Torch

Torch [30] — библиотека для научных вычислений с широкой поддержкой алгоритмов машинного обучения. Разрабатывается в Исследовательском институте Idiap при Нью-Йоркском университете, начиная

с 2000 года.

Библиотека реализована на языке Lua с использованием языка программирования C и программно-аппаратной архитектуры параллельных вычислений CUDA. Скриптовый язык Lua в совокупности с технологиями SSE, OpenMP, CUDA позволяют библиотеке Torch показывать более высокую скорость обработки сценариев по сравнению с другими библиотеками. На данный момент поддерживаются операционные системы Linux, FreeBSD, Mac OS X. Основные модули также работают и на Windows.

Библиотека состоит из набора модулей, каждый из которых отвечает за различные стадии работы с нейронными сетями. Так, например, модуль nn обеспечивает конфигурирование нейронной сети (определению слоев, и их параметров), модуль optim содержит реализации различных методов оптимизации, применяемых для обучения, a gnuplot предоставляет возможность визуализации данных (построение графиков, показ изображений и т.д.). Установка дополнительных модулей позволяет расширить функционал библиотеки.

Torch позволяет создавать сложные нейронные сети с помощью механизма контейнеров. Контейнер — это класс, объединяющий объявленные компоненты нейронной сети в одну общую конфигурацию, которая в дальнейшем может быть передана в процедуру обучения. Компонентом нейросети могут быть не только полносвязные или сверточные слои, но и функции активации или ошибки, а также готовые контейнеры. Torch позволяет создавать следующие слои:

- полносвязный слой;
- функции активации: гиперболический тангенс, выбор минимального или максимального, softmax-функция и другие;
- сверточные слои: свертка, прореживание, пространственное объединение, разностная нормализация.

Функции ошибки: среднеквадратичная ошибка, кросс-энтропия.

При обучении могут использоваться следующие методы оптимизации:

- стохастический градиентный спуск;
- усредненный стохастический градиентный спуск;
- алгоритм Бройдена-Флетчера-Гольдфарба-Шанно;
- метод сопряженных градиентов.

3.5.3 Фреймворк Caffe

Caffe [31] — это фреймворк глубокого машинного обучения, нацеленный на простое использование, высокую скорость и модульность.

Архитектура располагает к удобному использованию и расширению функционала. Модели определяются в файлах конфигурации и не требуют знаний в программировании. Переключение между центральным процессором (CPU) и графическим процессором (GPU) в настройках устанавливаются всего одним флагом и позволяет запустить обучение на GPU (что значительно быстрее), затем возможно размещение обученной модели на серверных кластерах или мобильных устройствах.

Производительность делает фреймворк Caffe инструментом пригодным для высокотребовательных экспериментов и позволяет обрабатывать до 60 миллионов изображений в день на одном графическом процессоре NVIDIA K40. То есть 1 миллисекунда на картинку для расчета и 4 миллисекунды на обучение.

Фреймворк Caffe используется в научно-исследовательских работах, стартапах, крупных промышленных приложениях в компьютерном зрении, распознавании речи и мультимедиа. Caffe в основном используется в качестве источника предварительно обученных моделей, размещенных на его сайте.

3.5.4 Модель VGG-19

VGG-19 [32] — это архитектура сверточной нейронной сети. Вход сети представляет собой изображение размера 224 × 224 пикселя. Как во всякой

сверточной нейронной сети, изображение проходит через сверточные слои, обрабатываясь фильтром с рецептивным полем размером 3 × 3 пикселя. Это минимальный размер фильтра, способный фиксировать такие понятия как «влево/вправо», «вниз/вверх» и «центр». Шаг свертки имеет фиксированное значение, и равен 1. Подвыборка осуществляется пятью слоями подвыборки по максимальному значению, которые следуют за некоторыми сверточными слоями. Подвыборка по максимальному значению имеет ядро 2 × 2 пикселя. Все скрытые слои оснащены нелинейной ректификацией, необходимой для обучения. Слои сети VGG-19 представлены в таблице 1.

Таблица 1 — Модель VGG-19

Слой	Результат на выходе		
	Ширина, пикс	Высота, пикс	Количество каналов
Вход	224	224	3
Conv (3×3×64)	224	224	64
Conv (3×3×64)	224	224	64
Pool (2×2)	112	112	64
Conv (3×3×128)	224	224	128
Conv (3×3×128)	224	224	128
Pool (2×2)	56	56	128
Conv (3×3×256)	56	56	256
Conv (3×3×256)	56	56	256
Conv (3×3×256)	56	56	256
Pool (2×2)	28	28	256
Conv (3×3×256)	28	28	512
Conv (3×3×256)	28	28	512
Conv (3×3×256)	28	28	512
Pool (2×2)	14	14	512
Conv (3×3×512)	14	14	512
Conv (3×3×512)	14	14	512
Conv (3×3×512)	14	14	512
Pool (2×2)	7	7	512

Так как на вход сети принимаются изображения 224 × 224 пикселя, то

перед обучением сеть автоматически обрезает исходное изображение, до необходимого размера. Архитектура модели нейронной сети VGG-19 представлена на рисунке 18.

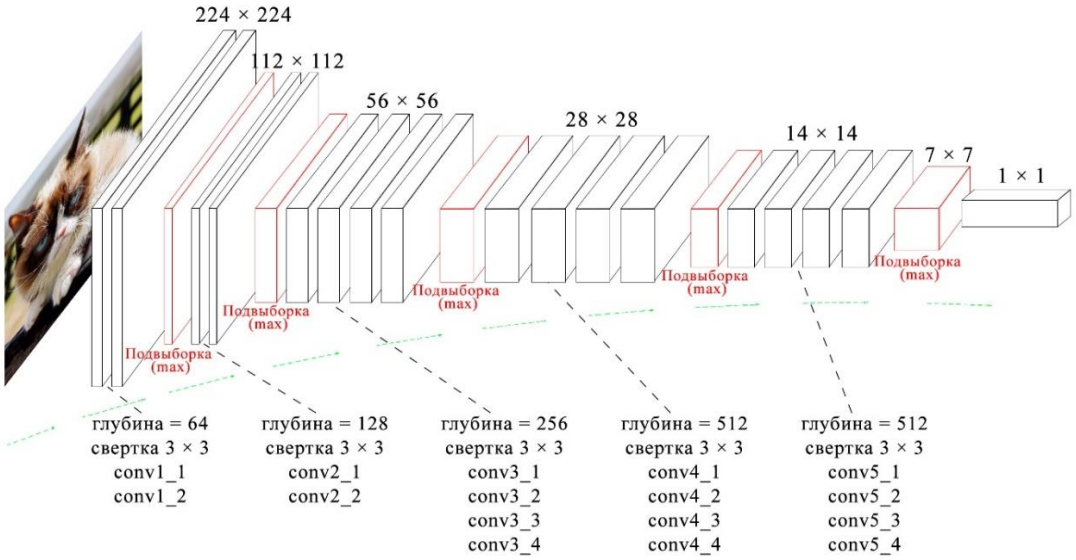

Рисунок 18 — Архитектура модели VGG-19

3.6 Реализация программного обеспечения

Для реализации была использована искусственная нейронная сеть, которая обеспечивает разделение содержимого от стиля, что позволяет перерисовывать содержимое одного изображения в стиле другого изображения. Для демонстрации работы, создавались художественные изображения, которые сочетают в себе стиль некоторых известных картин с содержимым произвольно выбранной фотографии.

Результаты, представленные в тексте выше, были сформированы на основе сверточной нейронной сети архитектуры VGG-19, разработанной для решения задач классификации изображений. Были использованы пространства признаков из 16 свёрточных слоев и 5 слоев подвыборки. В виду нецелесообразности не использовался ни один полносвязный слой нейронной сети. Модель VGG-19 общедоступна и может быть использована на

фреймворке для глубокого обучения нейронных сетей Caffe.

Первым шагом к реализации программы стала установка библиотеки для научных вычислений Torch.

Следующим шагом стала установка Loadcaffe - это модуль, загружающий модели Caffe в Torch. Далее были загружены модели VGG-19, предварительно обученной нейронной сети.

На языке Lua был написан код задающий конфигурацию нейронной сети, состоящей из таких параметров, как:

- изображение-стиль (style_image);
- изображение-содержимое (content_image);
- размер изображения (image_size);
- весовой коэффициент изображения-содержимого (content_weight);
- весовой коэффициент изображения-стиля (style_weight);
- тип слоя подвыборки (pooling);
- модель нейронной сети (model_file);
- слой(слои) для реконструкции содержимого (content_layers);
- слой(слои) для реконструкции стиля (style_layers).

Запуск программы осуществляется через терминал Linux, с помощью консольной команды. Во время обработки изображения, каждый 50 итераций, программа отображает промежуточные значения минимизируемой функции отклонения на слое содержимого, слоях стиля и ее глобальное значение, что отражено на рисунке 19.

Каждые 100 итераций минимизации программа генерирует промежуточный результат (см. приложение А, рисунок А. 1). Результат работы программы с различными изображениями стиля представлен в приложении А.

3.6 Вывод по главе 3

В данной главе представлены формы представления содержимого изображения и стиля изображения. Приведен алгоритм генерации изображений,

совмещающих в себе содержимое и стиль из разных источников. Итогом этого этапа явилось заключение о возможности применения различных представлений изображения для синтезирования новых на их основе.

Проведен выбор необходимых технологий для разработки информационной системы процедурной генерации изображений на основе искусственных нейронных сетей. Изучен вариант архитектуры искусственной нейронной сети, предназначенной для классификации изображений. Эта архитектура послужила основой для разработки программного обеспечения.

Разработан программный продукт, способный генерировать изображения, смешивая содержимое и стиль и разных исходных изображений. Приведены результаты опытной эксплуатации информационной системы.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы были изучены методы обработки изображений, произведен анализ современных алгоритмов и архитектур нейронных сетей. На основе предложенных методов и алгоритмов реализовано программное обеспечение для обработки изображений художественным стилем, с помощью нейронной сети, позволяющее генерировать изображение, которое смешивает содержимое изображения и стиль картины.

Представленные результаты экспериментальных исследований, приведенные в работе, могут вызвать интерес у представителей творческих профессий — графических дизайнеров, иллюстраторов, художников, мультипликаторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нейрохирурги с Ордынки [Электронный ресурс]. Режим доступа: https://www.weekit.ru/themes/detail.php?ID=78195.

2. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. — Москва: Горячая линия — Телеком, 2004. — 452 с.

3. Fyfe, C. Atificial neural networks / C. Fyfe. — Пейсли: The university of Paisley press, 1996. — 74 с.

4. Болотова, Ю.А., Спицын В.Г., Фомин А.Э. Применение модели иерархической временной памяти в распознавания изображений // Известия Томского политехнического университета. — 2011. — Т. 318. — № 5. — С. 60-63.

5. Anthony, M. Neural Network Learning: Theoretical Foundations : науч. изд. / M. Anthony, P. L. Bartlett. - Кембридж : Cambridge University Press, 1999. — 404 с.

6. Glorot, X. Deep sparse rectifier neural networks / X. Glorot // Fourteenth International Conference on Artificial Intelligence and Statistics. — 2011. — С. 315-323.

7. Neural Networks and Deep Learning [Электронный ресурс]. Режим доступа: http://neuralnetworksanddeeplearning.com.

8. Goodfellow I. Deep Learning : науч. изд. / I. Goodfellow, Y. Bengio, A. Courville. — Кембридж : MIT Press, 2016. — 800 с.

9. Хуршудов, А. А. Обучение многослойного разреженного автоэнкодера на изображениях большого масштаба / А. А. Хуршудов // Вестник компьютерных и информационных технологий. — 2014. — № 2. — С. 27-30.

10. Медведев, В. С. Нейронные сети MATLAB 6 / В. С. Медведев, В. Г. Потемкин. — Москва : ДИАЛОГ-МИФИ, 2002. — 496 с.

11. Фишер, Р. От поверхностей к объектам. Машинное зрение и анализ трехмерных сцен / Р. Фишер. — Москва: Радио и связь,1993. — 288 с.

12. Davies, E. R. Machine Vision: Theory, Algorithms, Practicalities / E. R. Davies. — Амстердам: Elsevier, 2014. — 572 с.

13. Bill Green. Canny Edge Detection Tutorial [Электронный ресурс] — Режим доступа: http://www.pages.drexel.edu/~weg22/can_tut.html.

14. Neelakanta, P. S. Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives / P. S. Neelakanta, D. DeGroff. — Бока-Ратон : CRC Press, 1994. — 256 с.

15. Гренадер, У. Лекции по теории образов: Регулярные структуры / У. Гренадер. — Москва : Мир, 1983. — 432 с.

16. Pitas, I. Digital Image Processing Algorithms and Applications / Pitas I. — Хобокен : John Wiley & Sons, 2000. — 419 с.

17. Бакут П. А., Колмогоров Г. С. Сегментация изображений: методы выделения границ областей // Зарубежная радиоэлектроника. — 1987. — № 5. — С. 25-47.

18. Фу, К. Структурные методы в распознавании образов / К. Фу. — Москва: Мир, 1977. — 320 с.

19. Галушкин А. И. Нейрокомпьютеры в биометрических системах / А. И. Галушкин. — Москва: Радиотехника, 2007. — 192 с.

20. Сыслов В. В. Нейросетевая система распознавания иероглифов // Нейрокомпьютер научно-технический журнал. — 1995. — № 2. — С. 13-21.

21. Ostagram [Электронный ресурс]. Режим доступа: https://ostagram.ru.

22. Deepart.io — become a digital artist [Электронный ресурс]. Режим доступа: https://deepart.io.

23. Prisma: Photo Editor, Art Filters Pic Effects [Электронный ресурс]. Режим доступа: https://itunes.apple.com/ru/app/prisma-photo-editor-art-filters-pic-effects/id1122649984?l=en&mt=8.

24. Malevich [Электронный ресурс]. Режим доступа: mlvch.com.

25. Understanding Deep Image Representations by Inverting Them [Электронный ресурс]. Режим доступа: https://arxiv.org/abs/1412.0035.

26. A Parametric Texture Model Based on Joint Statistics of Complex

Wavelet Coefficients [Электронный ресурс] — Режим доступа: http://link.springer.com/article/10.1023/A%3A1026553619983

27. Neural Algorithm of Artistic Style [Электронный ресурс] — Режим доступа: https://arxiv.org/abs/1508.06576.

28. Strothotte, T. Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation : науч. изд. / T. Strothotte, - Сан-Франциско: Morgan Kaufmann, 2002. — 496 с.

29. The Programming Language Lua [Электронный ресурс]. Режим доступа: https://www.lua.org.

30. Torch | Scientific computing for Lua [Электронный ресурс]. Режим доступа: http://torch.ch.

31. Caffe | Deep Learning Framework [Электронный ресурс]. Режим доступа: http://caffe.berkeleyvision.org.

32. Very Deep Convolutional Networks for Large-Scale Image Recognition [Электронный ресурс]. Режим доступа: https://arxiv.org/abs/1409.1556.

ПРИЛОЖЕНИЕ А

Демонстрационный материал

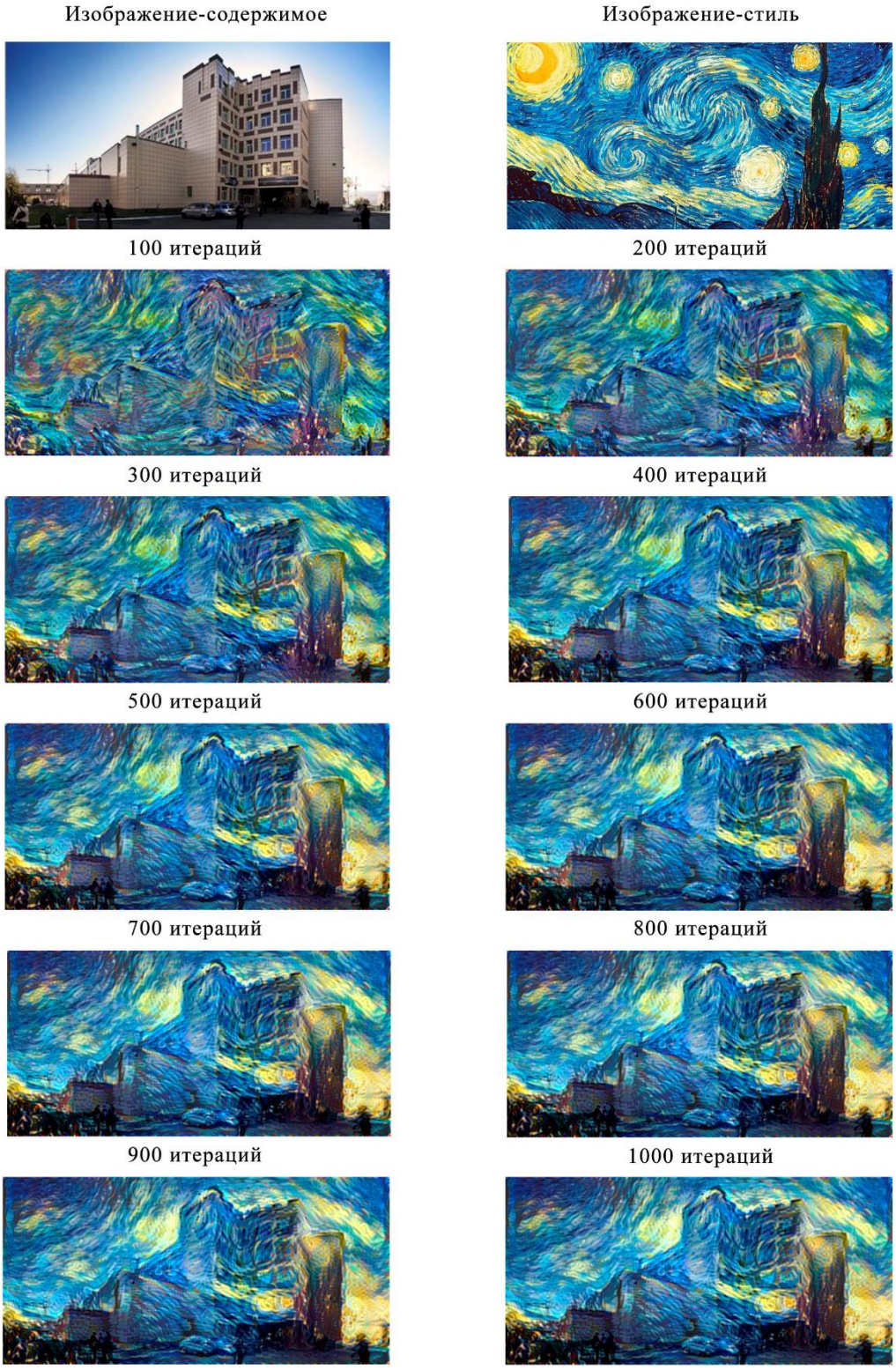

Рисунок А. 1 — Проежуточные результаты работы информационной системы процедурной генерации изображений с помощью нейронных сетей

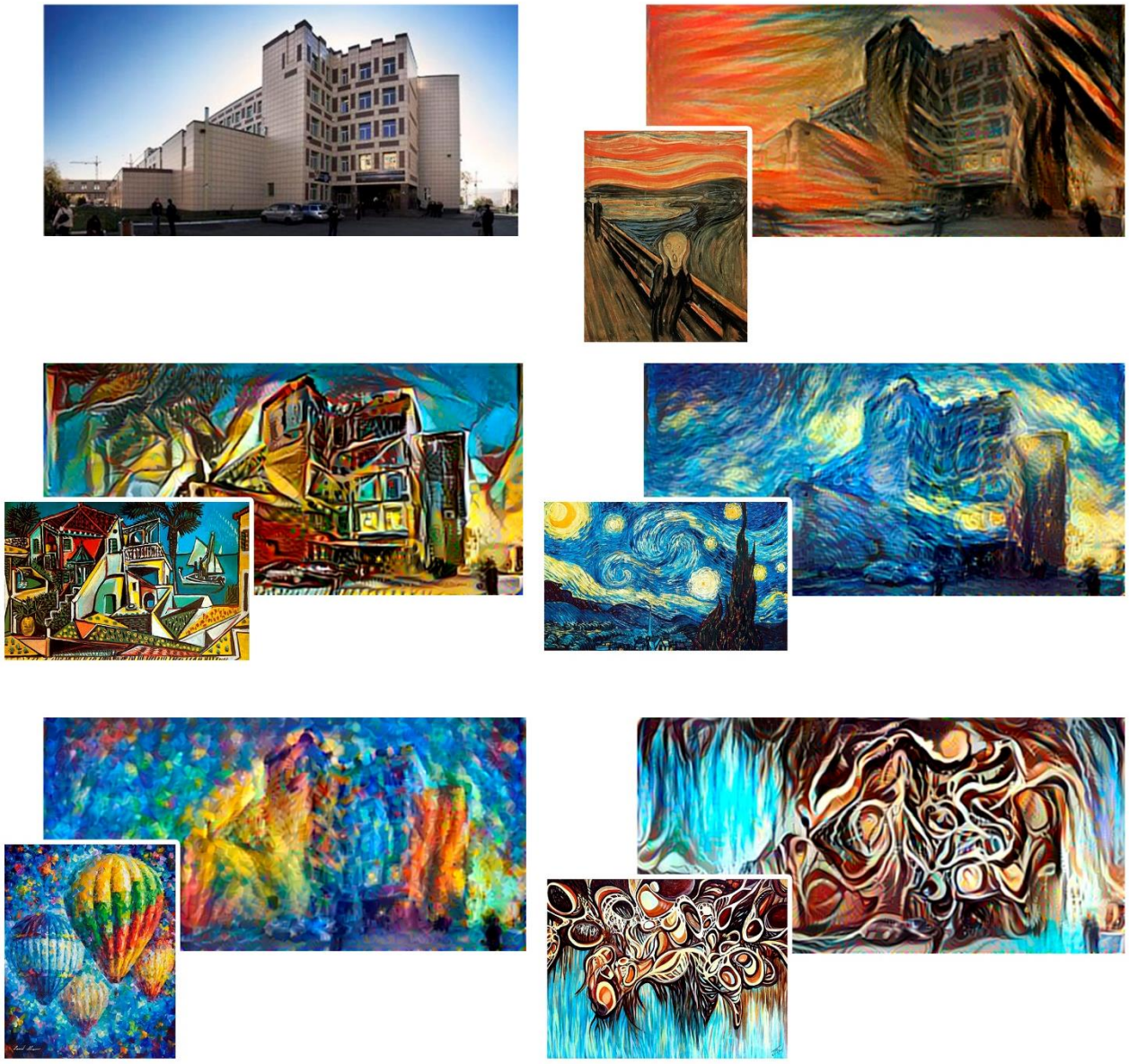

Рисунок А. 2 — Результаты работы информационной системы процедурной генерации изображений

Рисунок А. 3 — Результаты работы информационной системы процедурной генерации изображений

ПРИЛОЖЕНИЕ Б

Графический материал

Рисунок Б. 1 — Слайд презентации № 1

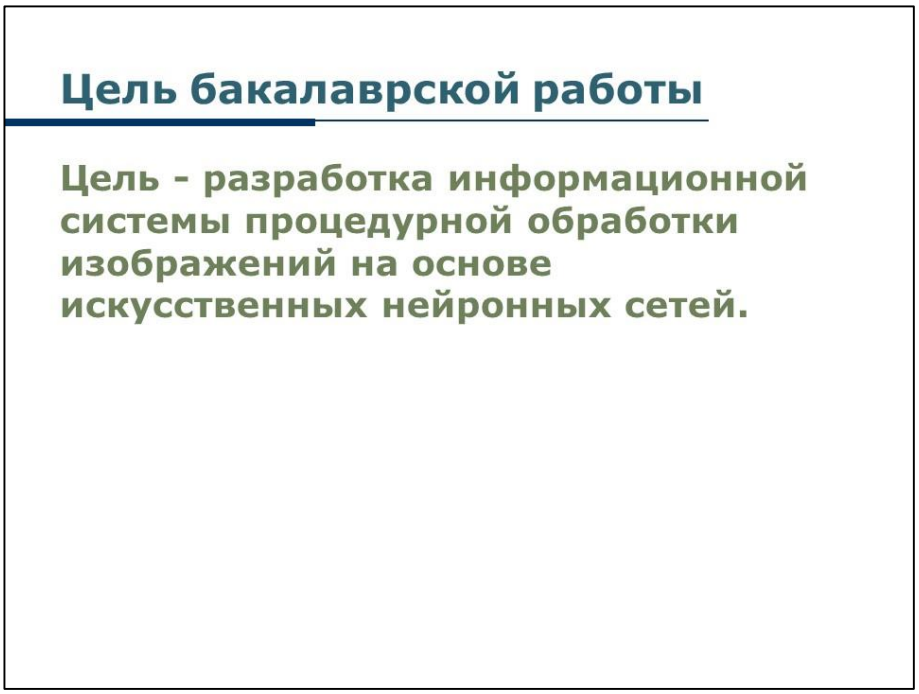

Рисунок Б. 2 — Слайд презентации № 2

Задачи бакалаврской работы

Для достижения поставленной цели необходимо было решить следующие задачи:

- **произвести анализ предметной области;**
- **изучить теоретические основы обработки изображений на основе нейронных сетей;**
- **разработать приложение процедурной обработки изображений на основе нейронных сетей;**
- **провести тестирование разработанной системы.**

Рисунок Б. 3 — Слайд презентации № 3

Почему нейронная сеть?

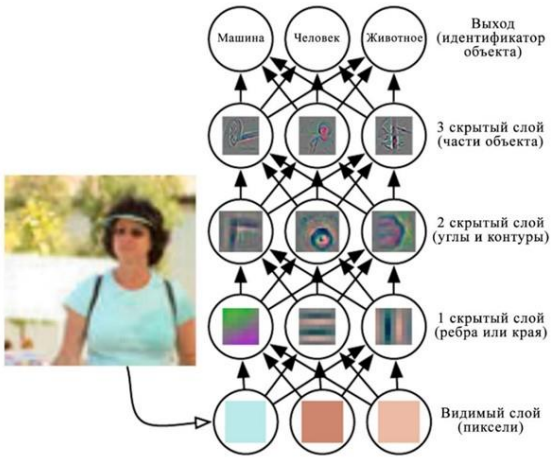

Рисунок Б. 4 — Слайд презентации № 4

Рисунок Б. 5 — Слайд презентации № 5

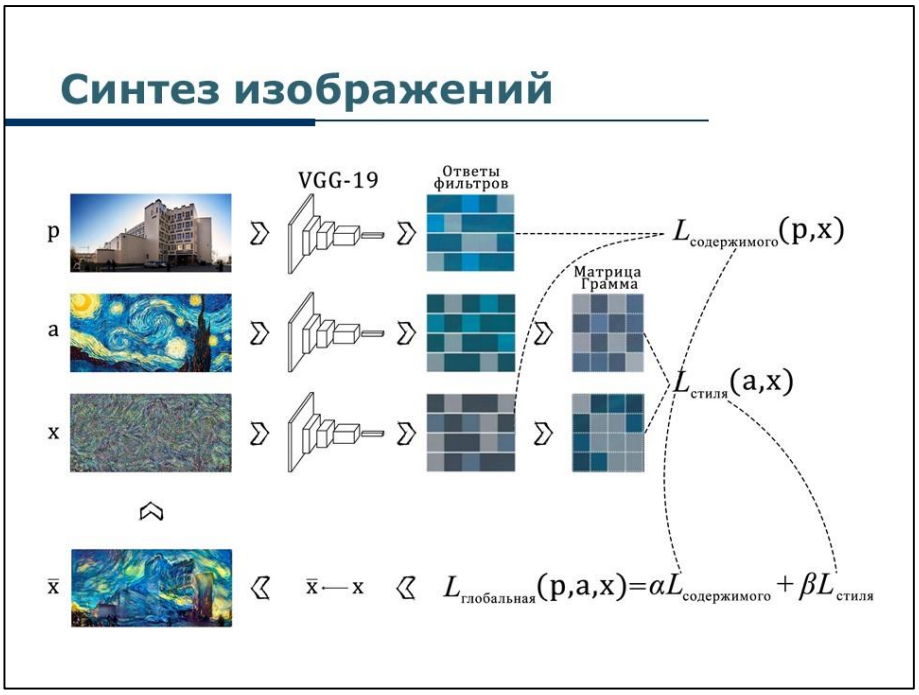

Рисунок Б. 6 — Слайд презентации № 6

Рисунок Б. 7 — Слайд презентации № 7

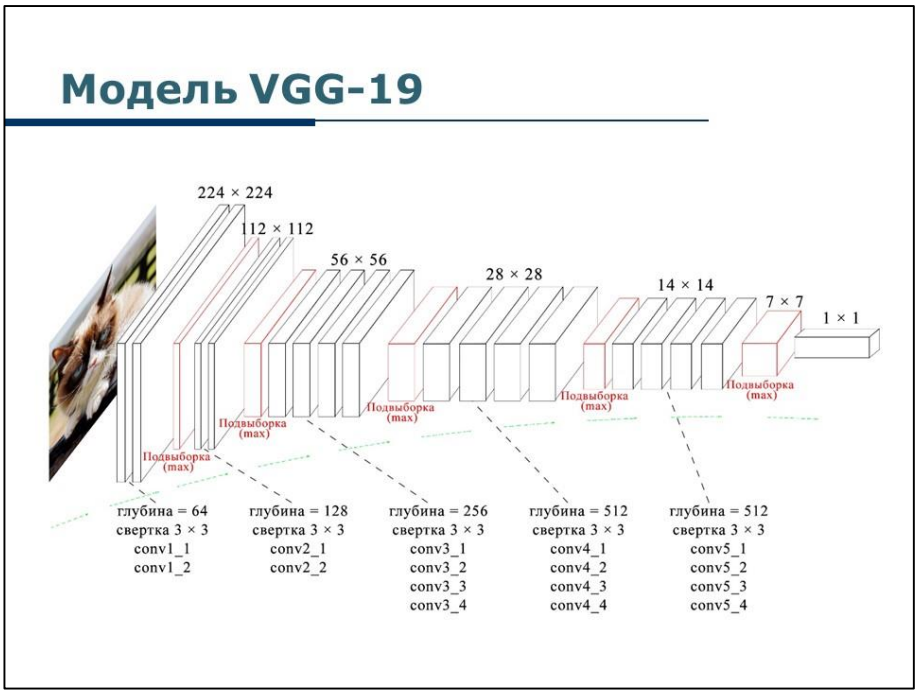

Рисунок Б. 8 — Слайд презентации № 8

Рисунок Б. 9 — Слайд презентации № 9

Рисунок Б. 10 — Слайд презентации № 10

Рисунок Б. 11 — Слайд презентации № 11

Рисунок Б. 12 — Слайд презентации № 12

Рисунок Б. 13 — Слайд презентации № 13

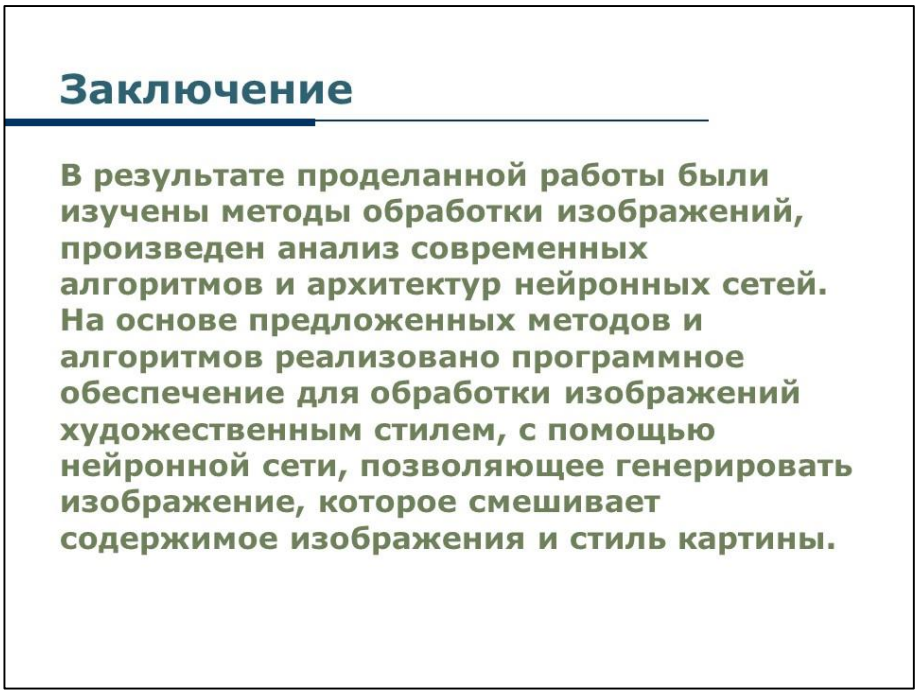

Рисунок Б. 14 — Слайд презентации № 14

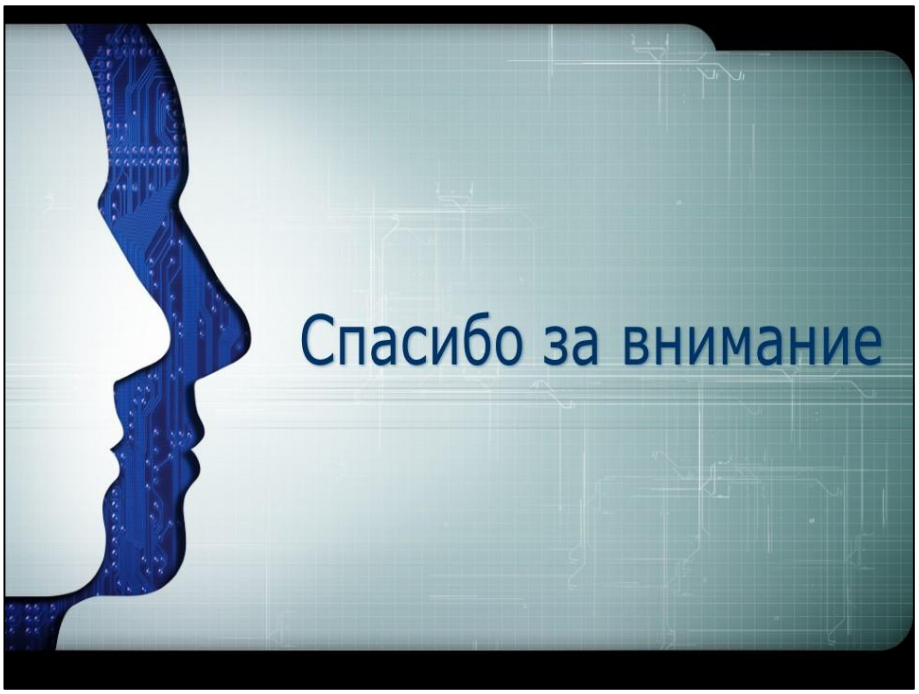

Рисунок Б. 15 — Слайд презентации № 15